VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Porovnání běhů SOMA algoritmu pomocí sítí

Comparisons of the SOMA Algorithm Runs by Means of Networks

2018 Patrik Lyčka

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Patrik Lyčka**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Porovnání běhů SOMA algoritmu pomocí sítí
Comparisons of the SOMA Algorithm Runs by Means of Networks

Jazyk vypracování: čeština

Zásady pro vypracování:

Biologicky inspirované algoritmy jsou založeny na nejrůznějších jevech v přírodě. Tyto algoritmy nám umožňují nalézt řešení problémů, které jsou klasickými metodami velmi obtížné či dokonce neřešitelné. Příkladem takovýchto algoritmů mohou být neuronové sítě, evoluční algoritmy, kolektivní inteligence či jiné. Pro pochopení jejich dynamiky můžeme využít například sítě (komplexní sítě). S pomocí sítí pak můžeme algoritmy analyzovat a následně algoritmy vylepšit.

Úkolem této práce je prostudovat oblast biologicky inspirovaných algoritmů, hlavně pak SOMA algoritmu, a teorie sítí, navrhnout převody tohoto algoritmu na síť a následně porovnat získané sítě pro různé běhy.

Cíle této práce je možné shrnout v těchto bodech.
1. Nastudovat a popsat problematiku biologicky inspirovaných výpočtů.
2. Nastudovat a popsat SOMA algoritmus.
3. Nastudovat a popsat problematiku sítí (komplexních sítí).
4. Naprogramovat SOMA algoritmus a jeho různé varianty.
5. Navrhnout a naprogramovat převody daného algoritmu na síť.
6. Program vhodně otestovat.
7. Porovnat vlastnosti získaných sítí pro různé běhy daného algoritmu - různé parametry, různé účelové funkce, lepší a horší nalezené řešení.

Seznam doporučené odborné literatury:

[1] BOCCALETTI, Stefano, et al. Complex networks: Structure and dynamics. Physics reports, 2006, 424.4: 175-308.
[2] NEWMAN, Mark. Networks: an introduction. 2010. United Slates: Oxford University Press Inc., New York
[3] ZELINKA, Ivan, et al. Evoluční výpočetní techniky: principy a aplikace. 1. vyd. Praha: BEN, 2009, 534 s. ISBN 978-80-7300-218-3.
[4] ZELINKA, Ivan. SOMA—Self-organizing Migrating Algorithm. In: Self-Organizing Migrating Algorithm. Springer International Publishing, 2016. p. 3-49.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Lukáš Tomaszek**

Datum zadání: 01.09.2017

Datum odevzdání: 30.04.2018

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 20. dubna 2018 .....

Rád bych na tomto místě poděkoval mému vedoucímu práce Ing. Lukáši Tomaszkovi za velmi cenné rady a ochotu kdykoliv pomoci. Dále bych chtěl také poděkovat mé rodině a přítelkyni za jejich trpělivost a podporu v mém studiu.

Abstrakt

Tato bakalářská práce se zaměřuje na analýzu algoritmu SOMA pomocí sítí. V této práci jsou obecně popsány biologicky inspirované algoritmy, evoluční algoritmy a detailně algoritmus SOMA. Také jsou zde popsány sítě (neorientované, orientované a vážené), metriky pro jejich analýzu a různé převody běhu SOMA algoritmu na síť. Dále v rámci této práce vznikl program pro analýzu sítí, který byl naiplementován a použit pro analýzu algoritmu SOMA. S využitím tohoto programu jsme porovnali běhy algoritmu SOMA pomocí metrik sítí pro různé funkce a lepší a horší nalezená řešení. Součástí práce je také uživatelská příručka pro tento program.

Klíčová slova: biologicky inspirované algoritmy, SOMA, komplexní sítě, metriky sítí, analýza běhů

Abstract

This bachelor thesis focuses on the SOMA algorithm analysis using networks. There are described biologically inspired algorithms, evolution algorithms and in detail the SOMA algorithm. Furthermore, this thesis deals with networks (undirected, directed and weighted), metrics for analyzing them and various SOMA algorithm conversions into the networks. Also, with this thesis there was created a tool for a network analysis. Using this tool, the different networks given from various SOMA algorithm runs were compared by measuring network metrics. Mainly we compared networks capturing best and worst runs and also we compared how the networks differentiate for the various cost functions. Additionally, the thesis contains the user manual for the tool.

Key Words: biologically inspired algorithms, SOMA, complex networks, network metrics, analysis of runs

Obsah

Seznam použitých zkratek a symbolů

CD	–	Compact Disc
DE	–	Differential Evolution
GUI	–	Graphical User Interface
INM	–	Interakční Nevážený Model
IRM	–	Interakční Rostoucí Model
IVM	–	Interakční Vážený Model
MM	–	Mravenčí Model
NES	–	Natural Evolution Strategy
PSO	–	Particle Swarm Optimalization
SOMA	–	Self-Organizing Migrating Algorithm
UML	–	Unified Modeling Language

Seznam obrázků

Seznam tabulek

1 Úvod

Biologicky inspirované výpočty [23] jsou třída algoritmů inspirovaných přírodními procesy. Jedná se o skupiny algoritmů, které umožňují efektivním způsobem řešit velmi složité problémy, jež jsou často klasickými metodami neřešitelné. Uplatnění naleznou v mnoha oblastech. Jako příklad lze uvést optimalizaci spotřeby paliva u leteckých motorů [33], simulaci pohybu tekutin [15], umělou inteligenci [35], detekci počítačových virů [12] nebo modelování růstu rostlin [16]. Do skupiny biologicky inspirovaných výpočtů spadají evoluční algoritmy a do těch patří algoritmus SOMA [33]. Běh tohoto algoritmu lze znázornit a analyzovat pomocí sítí a ze zjištěných informací pak algoritmus vylepšit [32].

Se sítěmi [17] jsme v kontaktu denně. Například při cestování automobilem (silniční sítě) [24], při surfování na internetu (počítačové sítě) [28], při používání elektrického proudu (elektrické sítě) [22] nebo při komunikaci s přáteli na Facebooku (sociální sítě) [27]. Sítě můžeme různými způsoby analyzovat a zjistit tak například nejvytěžovanější silnice ve městě [7] nebo průměrný počet přátel na Facebooku [27]. Informace pak můžeme využít například při plánování tras [10]. V této práci je budeme využívat pro analýzu algoritmu SOMA. V budoucnu pak můžou tyto informace vést k vylepšení tohoto algoritmu nebo k detekci úspěšného nalezení co možná nejlepšího řešení tímto algoritmem [32, 25].

V první kapitole si stručně popíšeme biologicky inspirované výpočty. Řekneme si, jaké podskupiny algoritmů zde spadají a jaké problémy řeší. V kapitole číslo 2 se zaměříme na podskupinu nazývající se evoluční algoritmy. Popíšeme si, čím jsou tyto algoritmy inspirovány, jaké mají společné rysy a uvedeme si pár konkrétních algoritmů z této podskupiny. V další kapitole se zaměříme přímo na algoritmus SOMA. Vysvětlíme si jak tento algoritmus funguje a jaké má vstupní parametry. Ve čtvrté kapitole si řekneme něco o testovacích funkcích, které budeme používat v této práci ve spojitosti s algoritmem SOMA. V kapitole číslo 5 se podíváme na sítě (grafy). Popíšeme si neorientované grafy, orientované grafy a vážené grafy, a jak lze tyto grafy v počítači reprezentovat. V následující kapitole se zaměříme na různé metriky grafu, co tyto metriky vyjadřují, a jak je lze spočítat. V osmé kapitole si popíšeme několik různých způsobů převodu běhu SOMA algoritmu na graf a význam jednotlivých metrik v těchto modelech. V kapitolách 9 a 10 si ukážeme, jak jsou algoritmus SOMA a program pro analýzu grafů naimplementovány, jak je nainstalovat a spustit, a jak je používat. V poslední kapitole se podíváme na vypočtené metriky při lepších a horších nalezených řešeních. Řekneme si, jak se liší a co z toho vyplývá pro algoritmus SOMA.

V této práci jsou použity anglické názvy funkcí, metrik a parametrů z důvodu komplikovaného překladu.

2 Biologicky inspirované výpočty

Biologicky inspirované výpočty [23], jak už sám název napovídá, jsou výpočty inspirované biologickými jevy, přičemž se využívá poznatků z biologie, informatiky a matematiky. Biologicky inspirované výpočty často úzce souvisí s oblastí umělé inteligence a strojového učení. Jedná se o použití počítačů k modelování přírodních jevů, studii života a ke zlepšení efektivity využití počítačů. Patří zde mnoho různých oblastí jako například:

1. Evoluční algoritmy [33] - Slouží pro hledání optimálního řešení v obrovském prostoru řešení. Jsou inspirovány evoluční teorií.

2. Celulární automaty [30] - Používají se pro modely systémů skládajících se z velkého počtu jednoduchých, stejných a vzájemně propojených komponent. Využívají se například pro simulování pohybu tekutin nebo zrnitých látek.

3. Neuronové sítě [35] - Uplatnění naleznou například při rozpoznávání a kompresi obrazu a zvuku [6, 11]. Jsou inspirovány neurony v mozku.

4. Umělé imunitní systémy [9] - Využívají se například při detekci a eliminaci virů v antivirových softwarech nebo při detekci průniků do počítačových sítí. Jsou inspirovány imunitním systémem.

5. Lindenmayerův systém [21] - Slouží pro modelování růstu rostlin [16], ale také pro generování různých křivek, fraktálů nebo pro modelování buněčných organismů. Je inspirovaný právě růstem rostlin v přírodě.

3 Evoluční algoritmy

Evoluční algoritmy [33] jsou inspirovány vývojem jedinců v přírodě. Jsou inspirovány Darwinovou evoluční teorií [8], která spojuje myšlenku postupné evoluce druhu s přirozeným výběrem, jakožto příčinou a hybnou silou evoluce.

Z rodičů jsou plozeni potomci, kteří při svém vzniku podléhají mutacím. Rodiče a potomci nevhodní pro aktuální životní prostředí pravidelně vymírají po generacích. Aby byla evoluce funkční, jsou nezbytné tři věci:

1. Je nezbytné umět z existujících řešení vytvořit nové, čemuž se říká křížení.
2. Je potřeba vytvořené řešení náhodně pozměnit, čemuž se říká mutace.
3. Musí se pro daného jedince vybrat vhodný jedinec ke křížení, čemuž se říká přirozený výběr.

Evoluční algoritmy patří mezi globálně optimalizační algoritmy, kdy je cílem prohledat prostor a najít optimální řešení. Ve většině případů je velikost prostoru obrovská, proto je nemožné prohledávat celý prostor řešení a právě tento problém řeší evoluční algoritmy. Lze je tedy využít například pro optimalizaci spotřeby paliva u leteckých motorů [33], nebo pro návrh elektrických obvodů [5], kde je velké množství možností, jak jednotlivé součástky zapojit. Cílem těchto algoritmů je v relativně rychlém čase dosáhnout optimálního řešení. Každý takovýto optimalizační problém si lze představit geometricky, kdy je cílem najít minimum, popřípadě maximum na N rozměrné ploše, která je obvykle zadána funkčním předpisem. Mezi evoluční algoritmy se řadí například DE [33] nebo NES [29].

Autor v publikaci [33] řadí mezi evoluční algoritmy také hejnové algoritmy. Hejnové algoritmy [31] jsou založeny na kolektivní inteligenci. Celý systém se skládá ze skupiny jedinců, z nichž každý řeší zadaný úkol sám, ale současně svůj postup v řešení sdílí s ostatními jedinci. Jedinci se tak navzájem ovlivňují. Hejnové algoritmy také slouží k řešení optimalizačních problémů. Jsou inspirovány například hejny ryb, ptáků nebo mravenců. Do skupiny těchto algoritmů patří například SOMA [33] nebo PSO [33].

4 SOMA

SOMA [33] se řadí mezi hejnové algoritmy. Podle publikace [33] ji lze zařadit také mezi evoluční algoritmy. Tento algoritmus se snaží napodobit chování inteligentních jedinců, kteří se společně snaží vyřešit nějaký problém.

Běh algoritmu SOMA je ovlivňován množinou parametrů, která musí být nastavena před začátkem běhu algoritmu. Nevýhodou je, že kvalita běhu algoritmu je závislá na nastavení těchto parametrů. V experimentální části této práce jsou tyto parametry nastaveny na doporučené hodnoty uvedené v tabulce 1.

Parametr	**Doporučená hodnota**
$PathLength$	1,1 až 5, dostatečná velikost je obvykle 3
$Step$	0,11 až PathLength
PRT	0 až 1, optimální hodnota je kolem 0,1
D	dáno problémem, počet argumentů účelové funkce
$PopSize$	minimálně 10
$Migrace$	minimálně 10
$MinDiv$	libovolný, záleží na hodnotách, jakých může optimalizovaná funkce nabývat

Tabulka 1: Doporučené hodnoty parametrů pro algoritmus SOMA

4.1 Parametry algoritmu

1. $PathLength$ – Tento paramer určuje, jak daleko bude aktivní jedinec cestovat, kdy hodnota $PathLength = 1$ je vzdálenost aktivního jedince od vedoucího jedince. Jedná se o relativní vzdálenost aktivního jedince od vedoucího jedince. Význam parametru je zobrazen na obrázku 1.

2. $Step$ – Určuje krok, s jakým bude mapována cesta aktivního jedince. Čím nižší $Step$ bude nastaven, tím podrobněji bude cesta prohledána. Je důležité nastavit $Step$ tak, aby vzdálenost mezi vedoucím a aktivním jedincem nebyla celočíselným násobkem parametru $Step$, protože by každý jedinec mohl být přitažen vedoucím jedincem. Proces by tak rychleji skončil v lokálním extrému a nemuselo by být nalezeno optimální řešení. Význam parametru je zobrazen na obrázku 1.

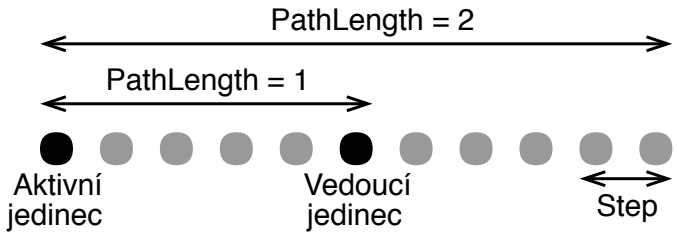
3. PRT – Podle tohoto parametru se tvoří $PRTVektor$, který ovlivňuje směr pohybu aktivního jedince k vedoucímu jedinci. Je to jeden z nejdůležitějších parametrů.

4. D – Jedná se o dimenzi účelové funkce. Je dána řešeným problémem.

5. $PopSize$ – Určuje počet jedinců v populaci.

6. Migrační kola – Udávají, kolikrát se bude populace přemisťovat neboli migrovat. Udávají tedy maximální počet cyklů (iterací).

7. *MinDiv* – Tento parametr určuje, jaký je povolen maximální rozdíl mezi nejhorším a nejlepším jedincem v aktuální populaci. Jestliže je rozdíl menší než tento parametr, je běh algoritmu ukončen. Pokud je tento parametr nastaven na příliš vysokou hodnotu, tak se algoritmus zastaví dříve, než se stihne lokalizovat extrém. Pokud je nastaven na příliš nízkou hodnotu, tak se algoritmus ukončí až s vyčerpáním maximálního počtu migračních kol.



Obrázek 1: Grafické znázornění významu parametrů *PathLength* a *Step*.

4.2 Princip algoritmu

Samotný algoritmus SOMA se skládá z několika kroků, které se neustále opakují. Nyní si jednotlivé kroky krátce popíšeme.

1. Definice parametrů – Před začátkem běhu algoritmu je nutno nadefinovat již zmíněné parametry a účelovou funkci.

2. Tvorba populace – V tomto kroku se vytvoří počáteční populace. Populace může být znázorněna jako matice $N \times M$, v níž řádky představují jednotlivé jedince a sloupce jednotlivé parametry jedince. Každý jedinec je množina argumentů účelové funkce, jejíž optimální kombinace je pomocí evolučního algoritmu hledána. S každým jedincem je navíc spojena hodnota účelové funkce, která říká, jak dobré řešení daný jedinec představuje. K vytvoření populace je zapotřebí nadefinovat vzorového jedince neboli specimen, podle kterého se generuje celá počáteční populace. V tomto vzoru je pro každý parametr jedince nadefinován datový typ proměnné a interval, v němž se může pohybovat hodnota parametru. Populace je poté vygenerována na základě vzorového jedince podle vztahu (1), který zajišťuje, že celá populace bude uvnitř prostoru možných řešení.

$$x_{i,j}^{(0)} = rnd[0..1] * (x_{i,j}^{(Hi)} - x_{i,j}^{(Lo)}) + x_{i,j}^{(Lo)} \tag{1}$$

$$i = 1, ..., M \qquad j = 1, ..., N$$

Index i označuje číslo jedince. Index j označuje parametr jedince. $x_{i,j}^{(0)}$ je vygenerovaný parametr j jedince i. $rnd[0..1]$ je náhodně vygenerované číslo z intervalu 0 až 1. $x_{i,j}^{(Hi)}$ je maximální hodnota parametru j jedince i. $x_{i,j}^{(Lo)}$ je minimální hodnota parametru j jedince i.

3. Migrační kola – Migrace zde představuje křížení. V tomto kroku je zvolen ten nejlepší jedinec, který se prohlásí za vedoucího. Následně se začnou všichni jedinci pohybovat k vedoucímu jedinci pomocí skoků daných parametrem *Step* a po každém skoku si každý jedinec přepočítá hodnotu účelové funkce. Tento pohyb jedinců po skocích pokračuje tak dlouho, jak určuje parametr $PathLength$. Směr tohoto pohybu je narušován pertubací. U tohoto algoritmu pertubace představuje mutaci. Pohyb jedinců v prostoru je tedy náhodně rušen. Sílu rušení pohybu jedinců určuje parametr PRT. Z tohoto parametru se poté generuje $PRTVektor$ zvlášť pro každého jedince, který je platný pouze pro jeden aktuální skok aktivního jedince. Při generování $PRTVektoru$ se pro každou jeho složku vygeneruje náhodné číslo z intervalu od 0 do 1 a porovná se s PRT parametrem. Pokud je vygenerované číslo menší než PRT parametr, pak je dané složce vektoru přiřazena 1 a pokud větší, tak je přiřazena 0. Pokud má $PRTVektor$ všechny složky rovny 1, tak aktivní jedinec putuje přímo k vedoucímu jedinci. Pokud má však nějaké složky rovny 0, tak se daná souřadnice nemění a zůstává stejná. Pohyb jedince se řídí rovnicí (2), kde počet hodnot, kterých může t nabýt je $PathLength/Step$.

$$x_{i,j}^{ML+1} = x_{i,j,start}^{ML} + (x_{L,j}^{ML} - x_{i,j,start}^{ML}) * t * PRTVektor_j \tag{2}$$

$$t \in < Step, PathLength >$$

$x_{i,j}^{ML+1}$ je nová hodnota parametru j jedince i po posunutí o krok t. $x_{i,j,start}^{ML}$ je původní hodnota parametru j jedince i. $x_{L,j}^{ML}$ je aktuální hodnota parametru j vedoucího jedince. $PRTVektor_j$ je hodnota parametru j pertubačního vektoru. Po ukončení běhu se jedinec vrací na pozici, kde byla nalezena nejlepší hodnota.

4. Testování ukončovacích parametrů – V tomto kroku se kontroluje, zda je rozdíl mezi vedoucím jedincem a nejhorším jedincem menší než parametr $MinDiv$ a také, zda je počet migračních kol menší než parametr $Migrace$. Další možností je kontrola počtu ohodnocení účelové funkce. Pokud ani jedna podmínka není splněna, tak se krok 3 opakuje.

5. Konec algoritmu – Algoritmus vrací nejlepšího nalezeného jedince a končí.

4.3 Různé varianty algoritmu

1. Všichni k jednomu – Tento druh byl popsán výše. Funguje na principu, ve kterém se všichni jedinci pohybují k tomu nejlepšímu.

2. Všichni ke všem – V této variaci se všichni jedinci nepohybují k tomu nejlepšímu, ale k sobě navzájem. Nové pozice jedinci zaujímají až po dokočení migrací všech jedinců. Tato metoda je výpočetně náročnější, ale prohledá větší část prostoru možných řešení.

3. Adaptivně všichni ke všem – Tato metoda funguje jako Všichni ke všem s tím rozdílem, že jedinec nezaujímá novou pozici až po dokončení migrací všech jedinců, ale přesouvá se po každé aktuálně dokončené migraci.

4. Všichni k jednomu náhodně – Funguje podobně jako Všichni k jednomu, s tím rozdílem, že vedoucí jedinec není určen nejlepší hodnotou účelové funkce, ale je pro migraci každého jedince náhodně zvolen.

5. Svazky – Tato úprava se dá použít na kterýkoliv předchozí druh algoritmu SOMA. Jedinci jsou rozděleni do podpopulací a v každé podpopulaci probíhá algoritmus SOMA samostatně.

5 Testovací funkce

Pro testování algoritmu jsme použili testovací funkce CEC 2014 [14]. Všechny testovací funkce jsou navrženy pro hledání minima. Pokud se nepřičte hodnota F_i^*, mají funkce miminum v 0. Každá funkce má prohledávací rozsah v intervalu $< -100, 100 >$ v každé dimenzi. Každá testovací funkce se skládá ze základní funkce a navíc je ještě případně rotována nebo posunuta. Vstupní data (rotační matice a posuny) jsou navrženy pro dimenze 2, 10, 20, 30, 50 a 100. Funkce jsou naimplementovány bez přičtení hodnoty F_i^* z důvodu snadnějšího zpracování výsledných dat. Přehled unimodálních funkcí je uveden v tabulce 2, multimodálních v tabulce 3, hybridních v tabulce 4 a kompozitních v tabulce 5.

Číslo	Název	$F_i^* = F_i(x^*)$
1	Rotated High Conditioned Elliptic Function	100
2	Rotated Bent Cigar Function	200
3	Rotated Discus Function	300

Tabulka 2: Přehled unimodálních testovacích funkcí CEC'14

Číslo	Název	$F_i^* = F_i(x^*)$
4	Shifted and Rotated Rosenbrock's Function	400
5	Shifted and Rotated Ackley's Function	500
6	Shifted and Rotated Weierstrass Function	600
7	Shifted and Rotated Griewank's Function	700
8	Shifted Rastrigin's Function	800
9	Shifted and Rotated Rastrigin's Function	900
10	Shifted Schwefel's Function	1000
11	Shifted and Rotated Schwefel's Function	1100
12	Shifted and Rotated Katsuura Function	1200
13	Shifted and Rotated HappyCat Function	1300
14	Shifted and Rotated HGBat Function	1400
15	Shifted and Rotated Expanded Griewank's plus Rosenbrock's Function	1500
16	Shifted and Rotated Expanded Scaffer's F6 Function	1600

Tabulka 3: Přehled multimodálních testovacích funkcí CEC'14

Číslo	Název	$F_i^* = F_i(x^*)$
17	Hybrid function 1 (N=3)	1700
18	Hybrid function 2 (N=3)	1800
19	Hybrid function 3 (N=4)	1900
20	Hybrid function 4 (N=4)	2000
21	Hybrid function 5 (N=5)	2100
22	Hybrid function 6 (N=5)	2200

Tabulka 4: Přehled hybridních testovacích funkcí CEC'14

Číslo	Název	$F_i^* = F_i(x^*)$
23	Composite function 1 (N=5)	2300
24	Composite function 2 (N=3)	2400
25	Composite function 3 (N=3)	2500
26	Composite function 4 (N=5)	2600
27	Composite function 5 (N=5)	2700
28	Composite function 6 (N=5)	2800
29	Composite function 7 (N=3)	2900
30	Composite function 8 (N=3)	3000

Tabulka 5: Přehled kompozitních testovacích funkcí CEC'14

6 Grafy

Grafů nebo jiným slovem sítí je mnoho typů. Od těch nejjednodušších, jako jsou neorientované grafy [17], až po složitější orientované vážené grafy [17]. Každý z těchto typů můžeme definovat a každý z nich nám dává jiné množství informací o problému, který popisuje.

6.1 Graf

Graf [17] je v té nejjednodušší formě množina bodů propojených navzájem křivkami. Tyto body se nazývají vrcholy a křivky hrany. Formálně, podle [13] graf G definujeme jako uspořádanou dvojici $G = (V, E)$, kde V je neprázdná množina vrcholů a E je množina hran - množina (některých) dvouprvkových podmnožin množiny V. Počet vrcholů grafu poté označujeme n.

Pomocí grafů můžeme znázornit mnoho procesů, například z oblastí fyziky, biologie nebo sociologie [17]. Můžeme pomocí nich znázornit například Internetové sítě [20] (síťová zařízení propojené pomocí různých přenosových médií), lidskou společnost [20] (lidi propojené navzájem v případě, že se znají) nebo silniční síť [24](křižovatky propojené silnicemi). Takový graf pak můžeme analyzovat a tím získat odpovědi na nejrůznější otázky, jako třeba: Jaká je největší vzdálenost mezi dvěma počítači v Internetu? Který člověk zná nejvíce lidí? Jaká je nejkratší cesta z Prahy do Ostravy?

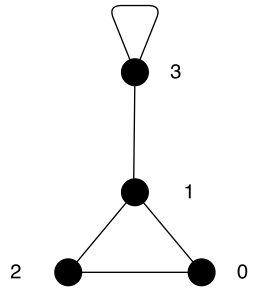
6.2 Matice sousednosti

Graf můžeme reprezentovat mnoha způsoby. Jedním z těchto způsobů je matice sousednosti. Tato matice je definována jako:

$$A_{ij} = \begin{cases} 1 \text{ pokud existuje hrana mezi vrcholy } i \text{ a } j. \\ 0 \text{ pokud neexistuje hrana mezi vrcholy } i \text{ a } j. \end{cases} \tag{3}$$

Matice sousednosti pro neorientovaný graf (graf, ve kterém hrany nemají směr) je symetrická a v případě, že graf neobsahuje smyčky (hrany vedoucí ze stejného do stejného vrcholu), má na diagonále pouze nuly. Pokud má nějaký vrchol smyčku, nastaví se v matici na patřičném místě číslo dvě. Dvojka proto, že smyčka (jako každá jiná hrana) má dva konce a oba jsou připojeny k témuž vrcholu. Příklad grafu a jeho reprezentaci pomocí matice sousednosti můžeme vidět na obrázku 2.

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 2 \end{pmatrix}$$



Obrázek 2: Neorientovaný graf a jeho matice sousednosti

6.3 Orientovaný graf

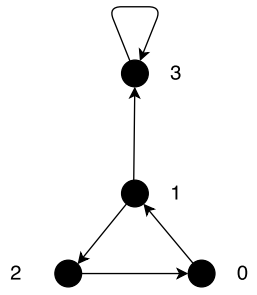
Orientované grafy jsou grafy, ve kterých má každá hrana navíc směr. Hrana tedy směřuje od jednoho vrcholu ke druhému. Takové hrany se nazývají orientované hrany. Formálně, podle [13] rozumíme orientovaným grafem G uspořádanou dvojici $G = (V, E)$, ve které je V množina vrcholů a $E \subseteq V \times V$ je množina orientovaných hran.

V případě silniční sítě, díky orientovaným hranám, můžeme znázornit například jednosměrné silnice. Tyto grafy opět můžeme reprezentovat pomocí matice sousednosti takto:

$$A_{ij} = \begin{cases} 1 \text{ pokud existuje hrana z vrcholu } j \text{ do } i. \\ 0 \text{ pokud neexistuje hrana z vrcholu } j \text{ do } i. \end{cases} \tag{4}$$

Matice sousednosti pro orientovaný graf již není symetrická. Pokud každou hranu v neorientovaném grafu nahradíme dvěma orientovanými hranami (každou v opačném směru) mezi stejným párem vrcholů, potom se na tento původně neorientovaný graf můžeme dívat jako na graf orientovaný. V případě smyček pak nastavujeme matici na patřičném místě pouze na 1 a ne na 2 jako tomu bylo u neorientovaného grafu. Příklad orientovaného grafu a jeho reprezentaci pomocí matice sousednosti můžeme vidět na obrázku 3.

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



Obrázek 3: Orientovaný graf a jeho matice sousednosti

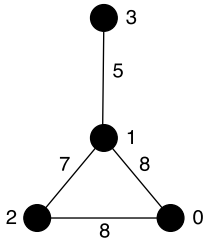
6.4 Vážený graf

Při popisu grafů si vždy nevystačíme pouze s informací o tom, zda mezi vrcholy hrana je nebo není. Někdy potřebujeme dané hraně přiřadit nějakou váhu (sílu), kterou obvykle reprezentujeme nějakým reálným číslem. V případě silniční sítě to může být například maximální povolená rychlost. V případě Internetu třeba přenosová rychlost daného propojení.

Formálně, podle [13] můžeme vážený graf definovat následovně. Ohodnocení grafu G je funkce $w : E(G) \to \mathbb{R}$, která každé hraně $e \in E(G)$ přiřadí reálné číslo $w(e)$, kterému říkáme váha hrany. Ohodnocený graf je graf G spolu s ohodnocením hran reálnými čísly.

Tento typ grafu můžeme také reprezentovat pomocí matice sousednosti a to tak, že dané číslo v matici bude reprezentovat váhu hrany. Příklad neorientovaného váženého grafu a jeho reprezentaci pomocí matice sousednosti můžeme vidět na obrázku 4.

$$A = \begin{pmatrix} 0 & 8 & 8 & 0 \\ 8 & 0 & 7 & 5 \\ 8 & 7 & 0 & 0 \\ 0 & 5 & 0 & 0 \end{pmatrix}$$



Obrázek 4: Neorientovaný vážený graf a jeho matice sousednosti

7 Metriky grafu

Z grafu můžeme vypočítat mnoho různých metrik, které nám o grafu dají různé informace. Pomocí metrik můžeme zjistit například důležitosti jednotlivých vrcholů, nejkratší vzdálenost mezi dvěma nejvzdálenějšími vrcholy v grafu, zda se podobné vrcholy mají tendenci navzájem spojovat a podobně.

7.1 Assortative mixing

Assortative mixing [19] nám říká, zda vrcholy v grafu mají tendenci se spojovat s jinými vrcholy, které jsou jim podobné. Jedna z možností jak Assortative mixing počítat je na základě stupňů vrcholů. To nám může dát například odpověď na otázku, zda se lidé s hodně přáteli mají tendenci přátelit s lidmi, kteří mají také hodně přátel. Assortative mixing pro orientovaný graf můžeme vypočítat takto:

$$r = \frac{\sum_i j_i o_i - M^{-1} \sum_i j_i \sum_{i'} o_{i'}}{\sqrt{[\sum_i j_i^2 - M^{-1}(\sum_i j_i)^2][\sum_i o_i^2 - M^{-1}(\sum_i o_i)^2]}}, \tag{5}$$

kde j_i je o 1 snížený vstupní stupeň vrcholu do kterého míří hrana i, o_i je o 1 snížený výstupní stupeň vrcholu ze kterého vychází hrana i a M je počet hran v grafu. Hodnota takto vypočteného Assortative mixing je v intervalu $< -1, 1 >$. Kladný výsledek znamená, že vrcholy mají tendenci se spojovat s podobnými vrcholy. Záporný výsledek znamená, že vrcholy mají tendenci se spojovat s odlišnými vrcholy. V případě neorientovaného grafu můžeme použít stejný vztah, ale v grafu nahradíme každou neorientovanou hranu dvěma orientovanými (každou v opačném směru).

7.2 Nejkratší cesta

Cesta v grafu [17] je taková sekvence vrcholů, kde jsou každé dva vrcholy v sekvenci spojeny hranou. V případě orientovaných grafů musí být každá hrana na cestě ve správném směru. Geodetická cesta nebo také nejkratší cesta je taková cesta mezi vrcholy, kdy žádná jiná kratší cesta již neexistuje. Délkou cesty se rozumí počet hran na cestě. V případě váženého grafu délka cesty odpovídá součtu ohodnocení hran na cestě.

7.3 Průměr grafu a průměrná délka cesty

Průměr grafu [17] je nejdelší cesta z nejkratších cest mezi vrcholy grafu.

Průměrná délka cesty je průměr délek nejkratších cest mezi všemi vrcholy grafu. Můžeme ji spočítat jako:

$$l_G = \frac{1}{n \cdot (n-1)} \cdot \sum_{i \neq j} d_{ij}. \tag{6}$$

7.4 Stupeň vrcholu

Stupeň vrcholu grafu (anglicky degree) [17] je počet připojených hran k vrcholu. V případě neorientovaného neváženého grafu s n vrcholy můžeme stupeň vrcholu z matice sousednosti spočítat jako:

$$k_i = \sum_{j=1}^{n} A_{ij}. \tag{7}$$

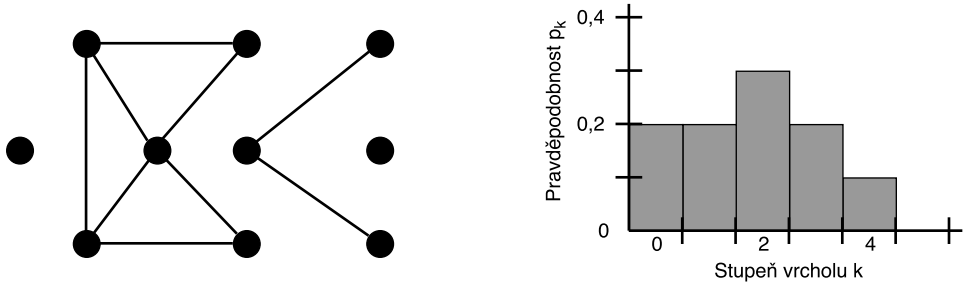
Stupeň vrcholu grafu v orientovaných grafech je složitější. V orientovaných grafech má každý vrchol dva druhy stupňů. Vstupní stupeň je počet vstupujících hran do vrcholu a výstupní stupeň je počet vystupujících hran z vrcholu. Z matice sousednosti v případě orientovaného neváženého grafu s n vrcholy můžeme stupně vrcholu vypočítat jako:

$$k_i^{in} = \sum_{j=1}^{n} A_{ij} \quad k_j^{out} = \sum_{i=1}^{n} A_{ij}. \tag{8}$$

Stupeň vrcholu ve vážených grafech [4] se poté označuje jako síla vrcholu a z matice sousednosti se vypočte stejným způsobem. Síla vrcholu spojuje informaci o počtu hran a váze jednotlivých hran připojených k vrcholu.

7.5 Degree distribution

Degree distribution [17] nám udává rozložení stupňů vrcholů v grafu. Představme si graf o 10 vrcholech, kde 2 vrcholy mají stupeň 0, 2 vrcholy stupeň 1, 3 vrcholy stupeň 2, 2 vrcholy stupeň 3 a 1 vrchol stupeň 4. Potom Degree distribution pro tento graf bude $p_0 = \frac{2}{10}$, $p_1 = \frac{2}{10}$, $p_2 = \frac{3}{10}$, $p_3 = \frac{2}{10}$, $p_4 = \frac{1}{10}$ a $p_k = 0$ pro všechna $k \geqq 5$. Hodnotu p_k můžeme také brát jako pravděpodobnost, která nám říká s jakou pravděpodobností bude mít náhodně vybraný vrchol grafu stupeň k. Pro přehlednost je potom vhodné Degree distribution zobrazit formou grafu funkce k, kterému se poté říká histogram. Histogram má obě osy logaritmické. Na obrázku 5 je ukázka Degree distribution jen ilustrační a osy v něm nejsou logaritmické.



Obrázek 5: Graf a jeho Degree Distribution

7.6 Clustering coefficient

Clustering coefficient [17] nám udává pravděpodobnost, že dva sousedé nějakého vrcholu jsou také sousedé (jsou také spojeni hranou). Tento parametr nám udává množství trojúhelníků v grafu. Clustering coefficient můžeme definovat jako:

$$K = \frac{(\text{počet trojúhelníků}) * 6}{\text{počet cest délky 2}}. \tag{9}$$

Clustering coefficient můžeme definovat také jinak. Pokud vrchol u je spojen s vrcholem v a vrchol v je spojen s vrcholem w, pak existuje v grafu cesta uvw, která má dvě hrany. Pokud vrchol u je také spojen s vrcholem w, pak říkáme, že je cesta uzavřená - obsahuje cyklus délky tři. Clustering coefficient poté můžeme vypočítat takto:

$$K = \frac{\text{počet uzavřených cest délky 2}}{\text{počet cest délky 2}}. \tag{10}$$

Clustering coefficient můžeme také definovat pro jeden vrchol. Poté se nazývá Lokální clustering coefficient a vyjadřuje pravděpodobnost, že náhodná dvojice vrcholů sousedících s daným vrcholem bude také navzájem sousedící (spojená hranou). Lokální clustering coefficient pro vrchol i můžeme spočítat jako:

$$K_i = \frac{\text{počet sousedních dvojic vrcholu } i \text{ jež jsou navzájem taky sousedící}}{\text{počet všech sousedních dvojic vrcholu } i}. \tag{11}$$

Clustering coefficient pro celý graf můžeme vypočítat také z lokálních clustering coefficientů takto:

$$K = \frac{1}{n} \sum_{i=1}^{n} K_i, \tag{12}$$

kde n je počet vrcholů v grafu.

7.7 Degree centralita

Centralita je jedním z důležitých parametrů grafu. Říká nám, jak moc je daný vrchol v grafu důležitý, případně jaký je nejdůležitější vrchol v grafu. Existuje mnoho způsobů, jak definovat centralitu, respektive důležitost vrcholu. Nejjednodušším druhem centrality je právě Degree centralita [17]. Tato centralita určuje důležitost vrcholu podle stupně vrcholu, tedy počtu hran připojených k vrcholu. V případě orientovaných grafů pak máme dva druhy této centrality - vstupní a výstupní. Degree centralitu můžeme spočítat takto:

$$C_i^D = k_i = \sum_{j=1}^{n} A_{ij}. \tag{13}$$

7.8 Eigenvector centralita

Tato centralita je rozšířením Degree centrality. Degree centralita bere v potaz pouze počet sousedních vrcholů, ale již ne jejich důležitost. Eigenvector centralita [17] vychází z toho, že důležitost vrcholu v grafu nezáleží jen na počtu vrcholů se kterými je vrchol spojen, ale také na důležitosti těchto vrcholů. Centralita každého vrcholu je tedy dána součtem centralit sousedních vrcholů. Z matice sousednosti můžeme Eigenvector centralitu C_i^E pro vrchol i, kde C'_j představuje centralitu souseda, spočítat takto:

$$C_i^E = \sum_j A_{ij} C'_j. \tag{14}$$

Tento přístup dává vysokou centralitu vrcholům, které mají hodně sousedů, nebo také vrcholům, které mají za souseda někoho s vysokou centralitou. Například z pohledu sociálních sítí může být důležitý člověk, který má kontakty na mnoho lidí nebo na pár lidí na vysokých místech. Tato centralita může být počítána jak pro neorientované tak pro orientované grafy.

7.9 Closeness centralita

Jiným způsobem měření centrality je Closeness centralita [17]. Tato centralita měří průměrnou vzdálenost z vrcholu do ostatních vrcholů. Předpokládejme, že d_{ij} je délka nejkratší cesty z vrcholu i do j. Potom průměrná vzdálenost z vrcholu i do všech ostatních vrcholů grafu je:

$$l_i = \frac{1}{n} \sum_j d_{ij}. \tag{15}$$

Closeness centralita pro vrchol i se poté vypočte jako:

$$C_i^C = \frac{1}{l_i} = \frac{n}{\sum_j d_{ij}}. \tag{16}$$

Tato centralita vychází z předpokladu, že vrcholy, které mají kratší průměrnou vzdálenost do ostatních vrcholů a mají tedy lepší přístup k informacím z ostatních vrcholů, jsou důležitější. Například v sociálních sítích, když má člověk kratší průměrnou vzdálenost k ostatním lidem, tak se jeho názor v komunitě šíří rychleji, než názory ostatních lidí s horší průměrnou vzdáleností.

7.10 Betweenness centralita

Tato centralita [17] měří množství nejkratších cest mezi všemi dvojicemi vrcholů, které procházejí daným vrcholem. Představme si síť, ve které putují informace po hranách z vrcholu do vrcholu, například datové pakety v Internetu. Vrcholy s vysokou Betweenness centralitou jsou poté schopny ovlivnit co nejvíce předávaných zpráv v síti. Tyto vrcholy jsou také ty, jejichž odebrání ze sítě bude mít největší dopad na ostatní vrcholy, protože přes ně vede nejvíce

cest, tedy přes ně putuje nejvíce zpráv. Tato centralita nám tedy popisuje vliv vrcholu na tok informací mezi ostatními vrcholy. Tuto centralitu pro vrchol i můžeme spočítat jako:

$$C_i^B = \sum_{st} n_{st}^i, \tag{17}$$

kde n_{st}^i označuje počet nejkratších cest vedoucích přes vrchol i z vrcholu s do t.

8 Převod běhu SOMA algoritmu na graf

Naimplementoval jsem několik převodů běhu SOMA algoritmu na graf, které níže popíšu. Převody se jmenují Interakční neválený model, Interakční válený model, Mravenčí model a Interakční rostoucí model. První tři vychází z publikace [26] a Interakční rostoucí model jsem navrhl sám.

8.1 Interakční neválený model

Představme si, že vytváříme graf zachycující běh algoritmu SOMA od migrace Y po migraci Z. Každý vrchol grafu představuje jednoho jedince, tudíž počet vrcholů grafu je roven parametru $PopSize$ algoritmu SOMA. Nechť jedince x_i představuje vrchol V_i a jedince x_j vrchol V_j. Pokud jedinec x_i při pohybu k jedinci x_j v migračním kole W, kde $Y \leqq W \leqq Z$, nalezl lepší řešení než své doposud nejlepší, tak je v grafu vytvořena orientovaná hrana z vrcholu V_i do vrcholu V_j. Pokud takováto orientovaná hrana z vrcholu V_i do vrcholu V_j již v grafu existuje, tak se již znovu do grafu nepřidává.

V tomto modelu mají metriky následující význam:

- Closeness centralita - Jak často byl daný jedinec vylepšen ostatními jedinci. Nejedná se jen o přímé vylepšení, ale také o vylepšení nepřímé po cestách.
- Betweness centralita - Jak často daný jedinec ovlivnil vylepšování ostatních jedinců, tedy byl vylepšen a předal informace dál.
- Eigenvector centralita - Jak často byl daný jedinec vylepšen důležitými jedinci v okolí.
- Stupeň vrcholu - Vstupní stupeň odpovídá počtu jedinců, kteří se vylepšili při pohybu k tomuto jedinci. Výstupní stupeň odpovídá počtu jedinců, o které se vylepšil při pohybu tento jedinec.
- Clustering coefficient - Jak často se jedinec x_i vylepšil o jedince x_j a x_k a zároveň se jedinec x_j také vylepšil o jedince x_k.
- Průměr grafu - Nízký průměr znamená, že se jedinci navzájem hodně vylepšovali. Vysoký průměr znamená, že se jedinci navzájem málo vylepšovali.
- Assortative Mixing - Zda se měli tendenci vylepšovat podobní jedinci, nebo rozdílní jedinci, případně tam nebyl sklon ani k jedné možnosti.

8.2 Interakční válený model

Tento model převodu je téměř stejný jako předchozí model. Jediným rozdílem je, že pokud máme přidat do grafu G orientovanou hranu z vrcholu V_i do vrcholu V_j a tato hrana již v grafu existuje, tak se zvýší váha této hrany o 1.

V tomto modelu mají metriky stejný význam jako v modelu předchozím, ale navíc se bere v těchto metrikách v úvahu také počet vylepšení. Další metrikou, která má v tomto modelu důležitý význam, je Strength. In-Srength odpovídá počtu vylepšení ostatních jedinců o jedince, pro kterého In-Strength počítáme. Out-Strength odpovídá počtu vylepšení tohoto jedince o ostatní jedince.

8.3 Mravenčí model

Tento model převodu je inspirován mravenci [34]. Když se mravenci pohybují, tak vypouštějí feromony. Tyto feromony se poté s přibývajícím časem postupně vypařují.

Tento model je modifikací modelu předchozího. Funguje úplně stejně, jen se navíc po skončení každého migračního kola (navýšení proměnné W) váha všech hran v grafu sníží o 1%, což představuje postupné zapomínání v síti. Pohyby jedinců v aktuálním migračním kole tak mají větší váhu, než pohyby jedinců o několik migračních kol zpět.

V tomto modelu mají metriky stejný význam jako v modelu předchozím, ale navíc se bere v těchto metrikách v úvahu také snížení váh hran z důvodu zapomínání v síti.

8.4 Interakční rostoucí model

Vytváříme graf, zachycující běh algoritmu SOMA od migrace Y po migraci Z. Každý vrchol grafu představuje jedince na nějaké konkrétní pozici (souřadnicích) v dané testovací funkci. Nejdříve tedy vytvoříme v grafu takový počet vrcholů, jako je počet jedinců v algoritmu SOMA. Nechť vrchol V_i představuje jedince x_i na pozici B_i a vrchol V_j jedince x_j na pozici B_j. Pokud jedinec x_i při pohybu k jedinci x_j v migračním kole W, kde $Y \leqq W \leqq Z$, nalezl lepší řešení než své doposud nejlepší, tak se po ukončení všech skoků daného migračního kola přesune na svou novou lepší pozici B_{i2}. Důsledkem toho se do grafu přidá vrchol V_{i2}, představující jedince x_i na pozici B_{i2}. Dále se vytvoří orientovaná hrana z vrcholu V_i do vrcholu V_{i2} a také z vrcholu V_j do vrcholu V_{i2}. Platí přitom, že pozice B_{i2} je lepší než pozice B_i (nemusí být lepší než pozice B_j). Tyto hrany vyjadřují, že interakcí daných dvou pozic byla objevena lepší pozice.

V tomto modelu mají metriky následující význam:

- Closeness centralita - Jak často jedinec na dané pozici přesouval ostatní jedince na nové pozice. Nejedná se jen o přímý přesun, ale také o přesun nepřímý po cestách.
- Betweness centralita - Čím vyšší betweness, tím vyšší podíl měl jedinec na dané pozici na přesunu ostatních jedinců na nové pozice.
- Eigenvector centralita - Jak často měl jedinec na dané pozici interakce s důležitými jedinci na důležitých pozicích v jeho okolí.
- Stupeň vrcholu - Vstupní stupeň vrcholu V_i je vždy roven 0 (jedinec na počáteční pozici) nebo 2 (jedinec na již přesunuté pozici). Výstupní stupeň vrcholu V_i vyjadřuje počet pozic, které byli navštíveny důsledkem interakce s jedincem x_i na pozici B_i.

- Clustering coefficient - Jak často vedoucí jedinec vylepšil nějakého jiného jedince a následně interakcí s tímto vylepšeným jedincem se opět jeden z nich vylepšil.
- Průměr grafu - Udává množství vedoucích jedinců během běhu algoritmu.

8.5 Váhy hran v grafech

Ve všech modelech, jejichž výsledkem je orientovaný vážený graf, se pro výpočet metrik, kde figuruje vzdálenost, jako třeba průměr grafu nebo closeness centralita, používá převrácená váha hrany. Například v silniční síti, kde váha představuje vzdálenost, je nižší váha lepší, protože to znamená, že vzdálenost je kratší. V případě SOMA algoritmu ale váha hrany znamená sílu vzájemné vazby. Silnější vazba, která odpovídá vyšší váze hrany je tak lepší. Z tohoto důvodu se ve výpočtech používá převrácená váha hrany [18].

9 Implementace

Naimplementoval jsem dva programy. První program je naimplementovaný algoritmus SOMA v různých verzích. Druhý program slouží pro analýzu běhu algoritmu SOMA pomocí různých metrik grafů.

9.1 Algoritmus SOMA

SOMU jsem naimplementoval v programovacím jazyce C++. V programu jsem využil knihovnu testovacích funkcí CEC 2014. Jedná se o konzolovou aplikaci, jejichž výstupem jsou log soubory, které následně dále využívá program pro analýzu grafů. Log soubory jsou textové soubory, jejichž jeden řádek má tento tvar:

```
migrační kolo;aktivní jedinec;vedoucí jedinec;step;původní hodnota aktivního
jedince; nová hodnota aktivního jedince;
```

Aplikaci je možné spustit z příkazového řádku a to s parametry v tomto pořadí:

Parametr	**Význam parametru**	**Ukázková hodnota**
PathLength	Jak daleko bude aktivní jedinec cestovat	3.1
Step	Krok s jakým bude aktivní jedinec cestovat	0.3
PRT	Míra pertubace pohybu jedince	0.3
Dimenze	Dimenze účelové funkce	10
PopSize	Počet jedinců v populaci	30
Migrace	Maximální počet cyklů	300
MinDiv	Přijatelný maximální rozdíl mezi nejlepším a nejhorším jedincem	0
Iterace	Počet spuštění algoritmu na každé funkci	500
Verze	1 = AllToOne, 2 = RandomAllToOne, 3 = AllToAll, 4 = AdaptiveAllToAll	1
OdFunkce	Od které CEC funkce spouštět simulace	1
PoFunci	Po kterou CEC funkci spouštět simulace	30

Tabulka 6: Parametry pro spuštění aplikace SOMA

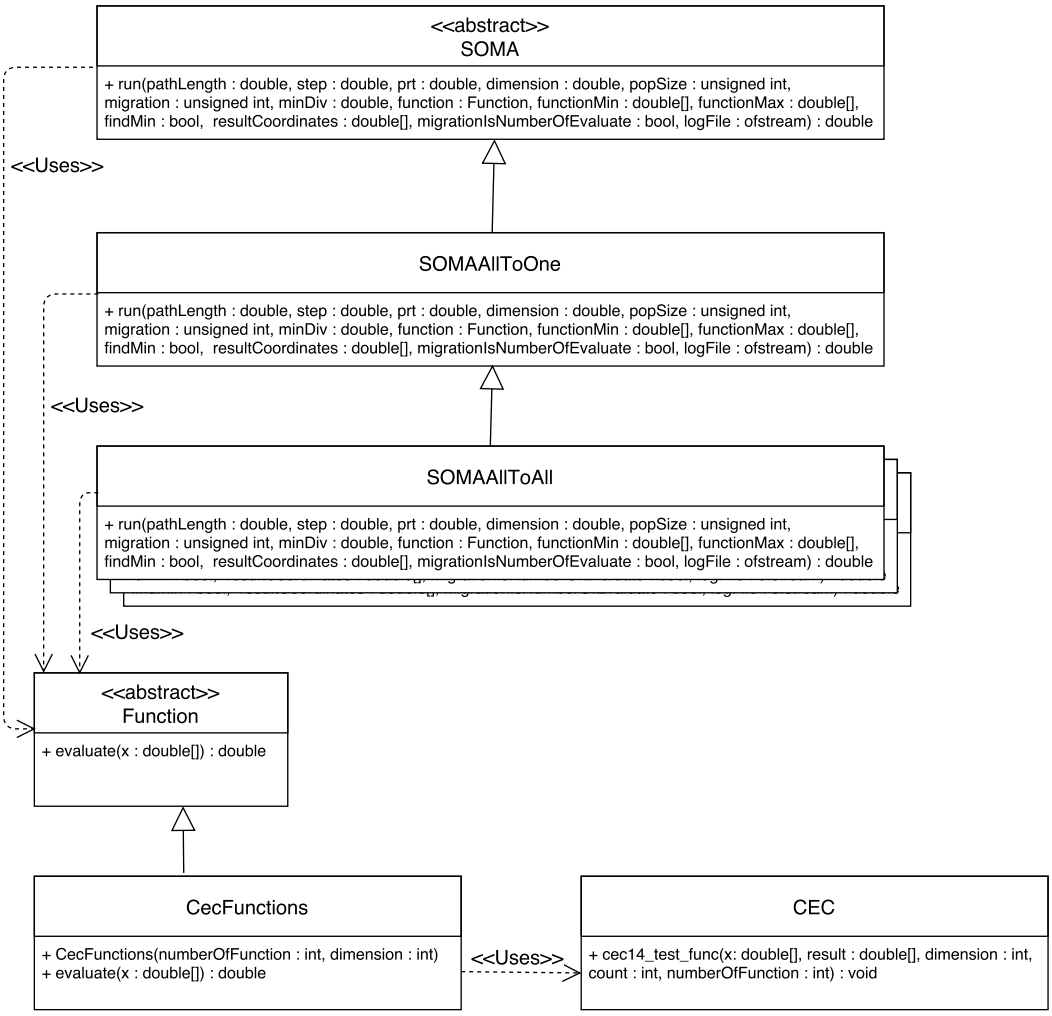
S ukázkovými parametry z tabulky 6 lze spustit aplikaci v operačním systému Windows následujícím příkazem:

```
soma.exe 3.1 0.3 0.3 10 30 300 0 500 1 1 30
```

Aplikace potřebuje pro svůj běh adresář input_data se vstupními soubory a adresář outputdata, kde se budou ukládat log soubory. Oba adresáře musí být ve stejném adresáři jako soubor

soma.exe. Datové soubory pro adresář input_data jsou staženy z webu [1]. Aplikace je k dispozici i s daty pro input_data v příloze na CD.

Aplikace je naimplementována podle následujícího UML diagramu zobrazeného na obrázku 6.



Obrázek 6: UML diagram algoritmu SOMA

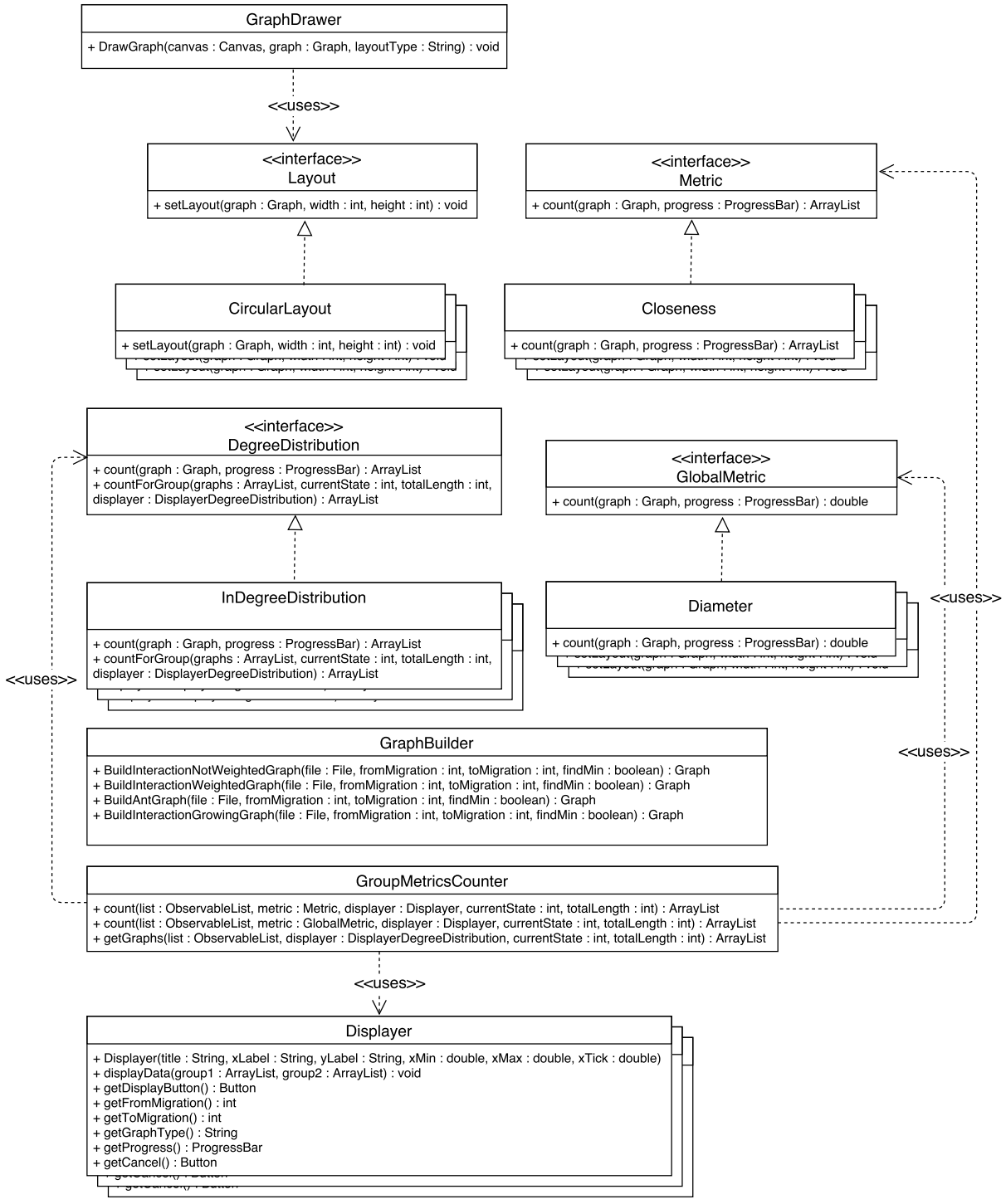
Abstraktní třída *SOMA* slouží jako vzor pro ostatní třídy, které představují jednotlivé verze tohoto algoritmu. Z této třídy poté dědí třída *SOMAAllToOne*. Všechny ostatní verze algoritmu dědí z třídy *SOMAAllToOne*. Účelovou funkci pro tento algoritmus poté přidáme pomocí parametru *function*, který je typu *Function*. Z této třídy dědí *CecFunctions*, která zajišťuje komunikaci s CEC funkcemi.

9.2 Program pro analýzu grafů

Program pro analýzu grafů jsem naimplementoval v programovacím jazyce Java. Při implementaci jsem využil knihovnu Jung [3], což je knihovna pro práci s grafy. GUI aplikace je vytvořeno s využitím technologie Java FX. Jako vstup program využívá log soubory z algoritmu SOMA.

Tyto soubory následně převádí na grafy, které je možné analyzovat. Výstupem programu jsou graficky znázorněné vypočtené metriky.

Aplikace je naimplementována podle následujícího UML diagramu zobrazeného na obrázku 7.



Obrázek 7: UML diagram programu pro analýzu grafů (nejdůležitější třídy)

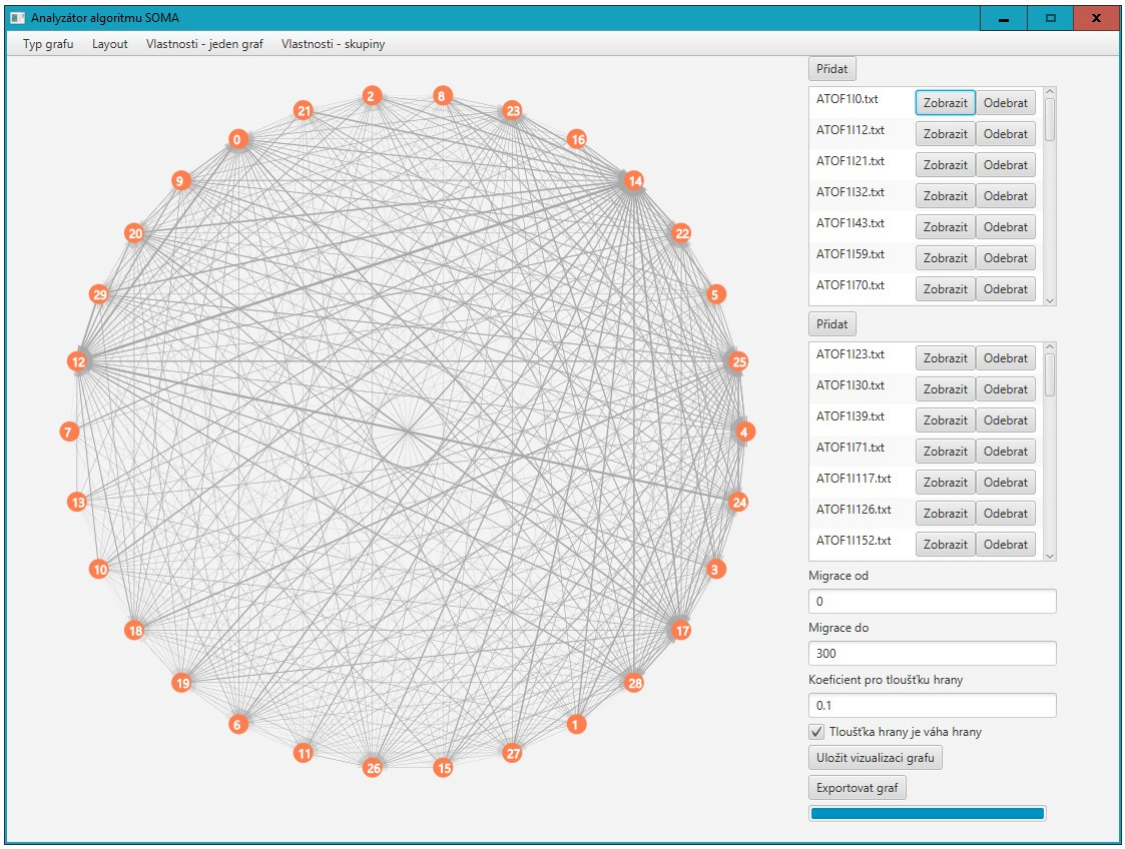
Jednotlivé třídy řeší následující úlohy:

- Třída *GraphDrawer* se stará o vykreslování struktury grafu. Pro rozmístění vrcholů na plátně využívá layouty z knihovny Jung. S těmito layouty komunikuje prostřednictvím rozhraní *Layout*.
- Třída *GraphBuilder* vytváří pomocí různých převodů graf (třídu *Graph* knihovny Jung) z log souborů, vytvořených algoritmem SOMA.
- Každá metrika je reprezentována jednou třídou. Některé metriky jsou vypočteny pomocí algoritmů naimplementovaných v knihovně Jung a některé jsem naimplementoval sám. Tyto třídy komunikují se zbytkem programu pomocí rozhraní *Metric*, *GlobalMetric* nebo *DegreeDistribution* podle toho o jaký typ metriky se jedná.
- Třída *GroupMetricsCounter* počítá metriku pro více grafů.
- Třídy skupiny *Displayer* zobrazují vypočtená data uživateli formou grafu. *Displayer* zobrazuje data pro dvě skupiny grafů, *DisplayerForOneGraph* data pro jeden graf, *DisplayerForOneGraphGlobal* globální metriky pro jeden graf, *DisplayerForOneGraphDegreeDistribution* degree distribution pro jeden graf a *DisplayerDegreeDistribution* degree distribution pro dvě skupiny grafů.

10 Manuál programu pro analýzu grafů

Program pro analýzu grafů se nachází v příloze na CD. Samotný program není třeba instalovat. Program stačí spustit pomocí souboru NetworkAnalyzerFX.jar. Aplikace je napsána v jazyce Java, tudíž pro spuštění programu je nutné mít Javu nainstalovanou v počítači. Javu lze stáhnout z webu [2].

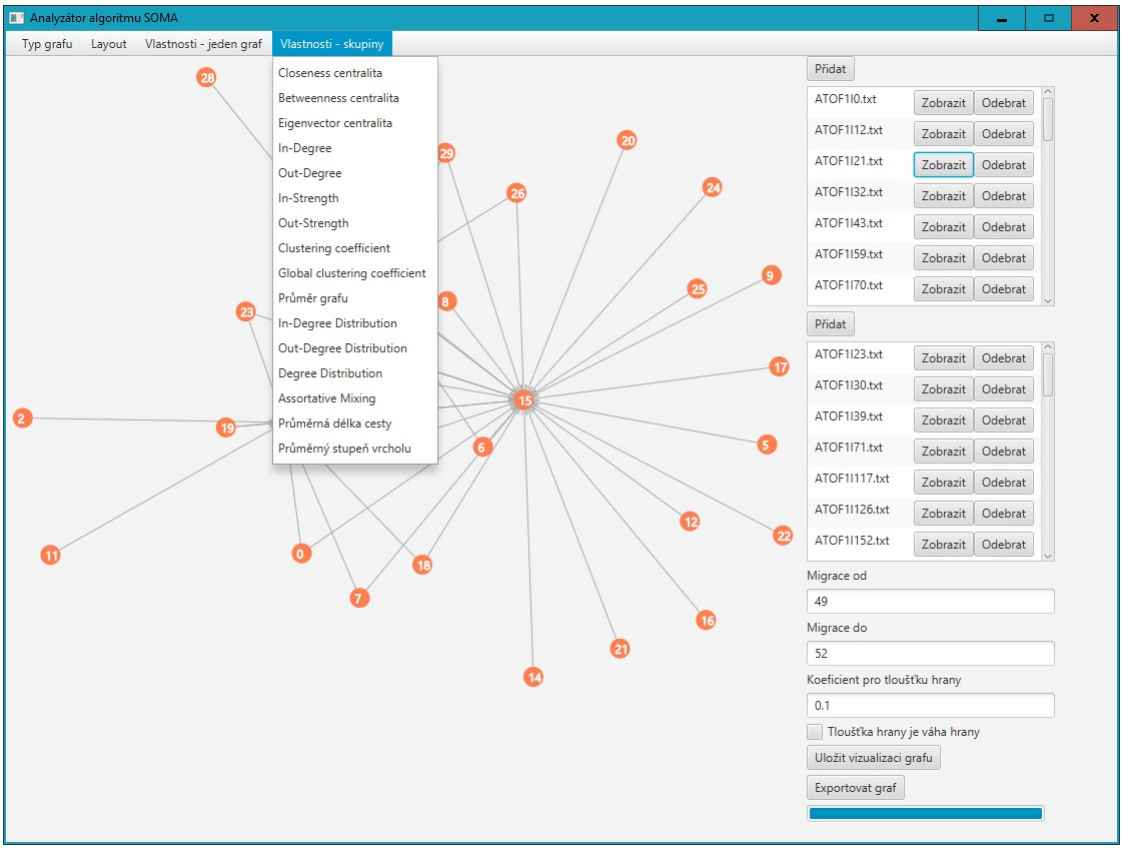
10.1 Ovládání programu



Obrázek 8: Ukázka úvodní obrazovky

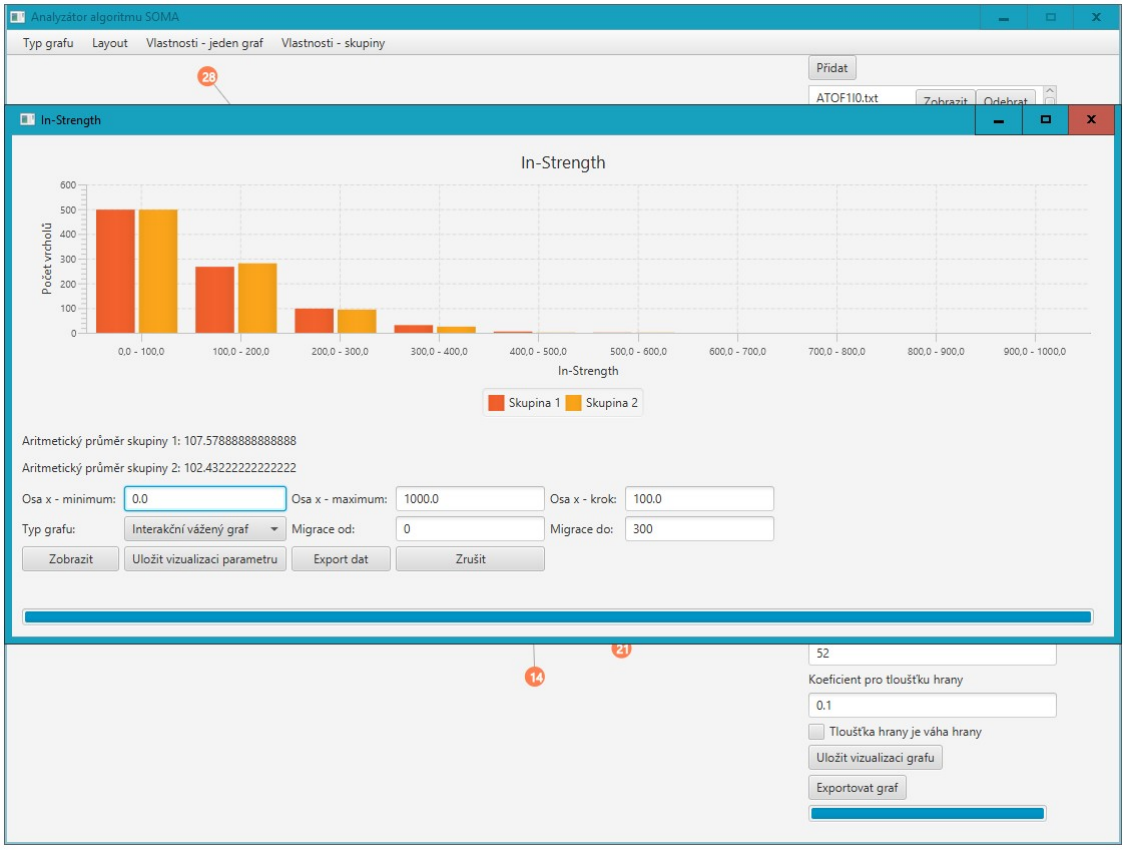
Po spuštění programu se zobrazí úvodní obrazovka (obrázek 8). V pravém sloupci této obrazovky se nachází dvě skupiny grafů (logy souborů). Do každé z těchto skupin se dá přidat další graf (případně několik grafů) pomocí tlačítka *Přidat* u dané skupiny nebo se dají grafy ze skupiny odstranit pomocí tlačítka *Odebrat* u příslušného grafu. V tomto okně se dá také zobrazit graf pomocí tlačítka *Zobrazit* u příslušného grafu. V pravém sloupci dole lze nastavit parametry, které ovlivňují vizualizaci grafu. Lze zde nastavit od kterého a po které migrační kolo algoritmu se bude graf vytvářet, koeficient pro tloušťku hrany v závislosti na váze hrany a lze vypnout zobrazování váhy hrany pomocí tloušťky hrany. Dále je možné pomocí tlačítka *Uložit vizualizaci*

grafu uložit graf do PNG souboru a pomocí tlačítka *Exportovat graf* uložit graf formou textového souboru.



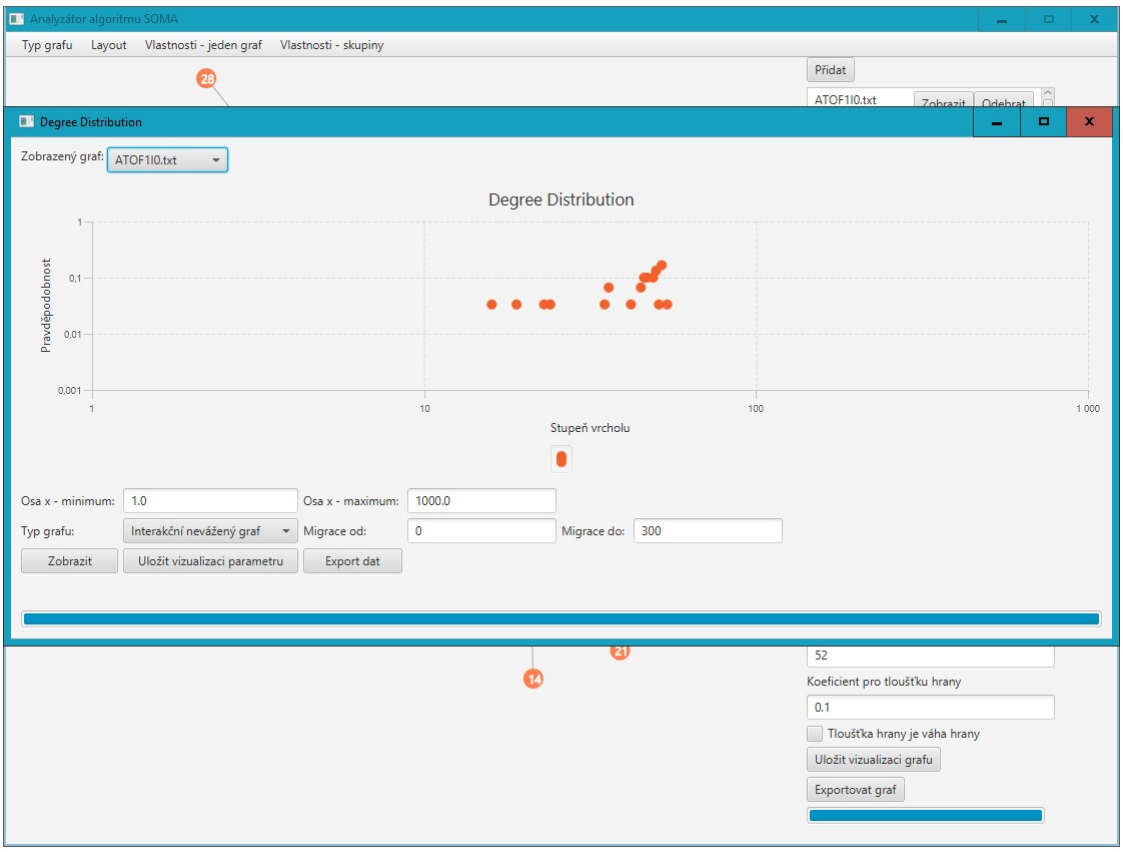
Obrázek 9: Ukázka úvodní obrazovky - volba metriky

V horní části úvodní obrazovky se nachází menu (obrázek 9). Položka *Typ grafu* v menu umožňuje zvolit typ grafu (model převodu), který se vytvoří z log souboru. Položka *Layout* poté určuje rozložení jednotlivých vrcholů grafu. Volba *Vlastnosti - skupiny* obsahuje několik různých metrik, které lze spočítat pro dané dvě skupiny grafů. Po zvolení jedné z metrik se zobrazí nové okno pro danou metriku (obrázek 10).



Obrázek 10: Ukázka metriky pro skupiny - In-Strength

V horní části tohoto okna se nachází graf, který znázorňuje vypočtené hodnoty dané metriky pro obě skupiny grafů a umožňuje tak obě skupiny mezi sebou vzájemně porovnat. Přímo pod grafem se nachází aritmetické průměry hodnot pro obě skupiny grafů. Pod těmito průměry se nachází několik parametrů, které ovlivňují jak zobrazení výsledků, tak jejich samotný výpočet. Můžeme zde nastavit minimální a maximální hodnotu osy x a také krok, s jakým se budou data zobrazovat. Dále zde můžeme nastavit typ grafu (model převodu), na kterém se bude výpočet metriky provádět a také rozsah migračních kol algoritmu, pro které se bude výpočet provádět. Po stisknutí tlačítka *Zobrazit* se následně spustí výpočet pro nastavené parametry. Průběh výpočtu zobrazuje indikátor průběhu ve spodní části okna. Tlačítkem *Zrušit* je možné tento výpočet předčasně ukončit. Vypočtené a zobrazené údaje lze pomocí tlačítka *Uložit vizualizaci parametru* uložit formou PNG obrázku grafu nebo pomocí tlačítka *Export dat* uložit formou textového souboru.



Obrázek 11: Ukázka metriky pro jeden graf - Degree distribution

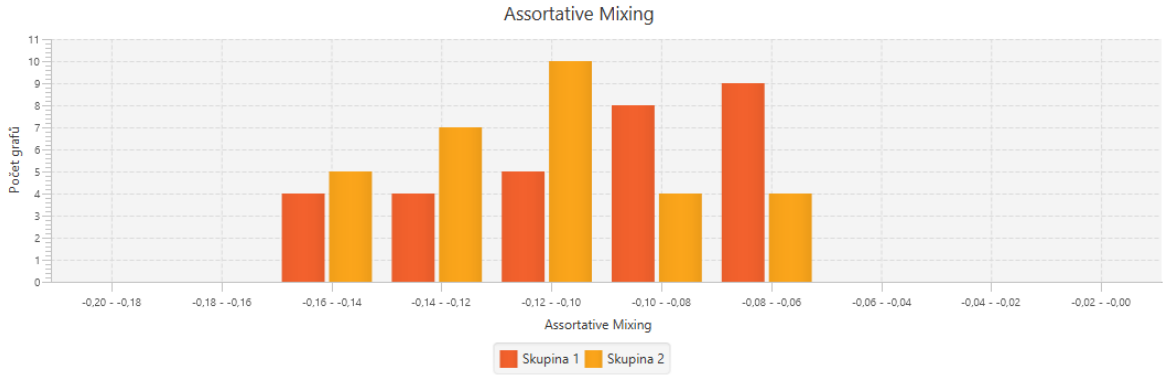
Pod volbou *Vlastnosti - jeden graf* v menu na úvodní obrazovce se nachází stejné metriky jako v předchozím případě, ale počítají se zde pouze pro jeden konkrétní graf a ne pro skupiny grafů. Po zvolení jedné z metrik se zobrazí nové okno pro danou metriku (obrázek 11). V horním levém rohu tohoto okna se nachází rozbalovací seznam pro volbu grafu, pro který se bude daná metrika počítat. Pod ním se opět nachází graf, který graficky znázorňuje vypočtené hodnoty dané metriky pro zvolený graf. Pod tímto grafem se opět nachází stejné parametry jako v předchozím případě u skupin grafů (v případě metriky Degree distribution na obrázku v ukázce chybí možnost krok, protože se jedná o logaritmické osy). Opět lze vypočtené hodnoty uložit formou PNG obrázku i textového souboru.

11 Analýza dat

Pro analýzu běhu SOMA algoritmu pomocí sítí jsme zvolili funkce Rotated High Conditioned Elliptic Function (F01) - unimodální, Shifted Schwefel's Function (F10) - multimodální, Hybrid function 4 (F20) - hybridní a Composite function 2 (F24) - kompozitní. Na těchto funkcích byl spuštěn SOMA algoritmus ve verzi AllToOne. Níže budou popsány nalezené rozdíly v různých metrikách a modelech. Skupina 1 (označená červeně) jsou úspěšnější běhy algoritmu (lépe nalezená řešení) a skupina 2 (označená oranžově) jsou méně úspěšné běhy algoritmu (hůře nalezená řešení). Všechny histogramy použité pro analýzu jsou k dispozici v příloze na CD.

11.1 Assortative Mixing

Assorative Mixing se jeví v obou skupinách napříč všemi modely a zvolenými funkcemi podobně. Vždy je mírně v záporných hodnotách, z čehož vyplývá, že jedinci v algoritmu mají tendenci úspěšně interagovat s mírně rozdílnými jedinci. Jediná závislost mezi úspěšnými a méně úspěšnými běhy je v případě unimodální funkce společně s INM. Z tohoto histogramu na obrázku 12 vyplývá, že běhy s lepším řešením dosahují častěji hodnot bližších k 0. Toto ale neplatí u ostatních analyzovaných funkcí.

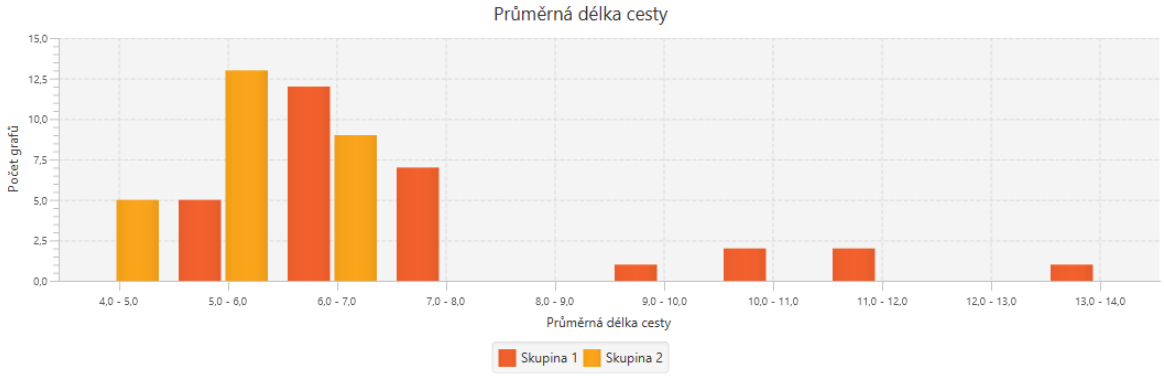


Obrázek 12: Rotated High Conditioned Elliptic Function - INM - Assortative Mixing

11.2 Průměrná délka cesty

V případě průměrné délky cesty je velký rozdíl mezi úspěšnými a méně úspěšnými běhy algoritmu. Úspěšnější běhy mají vždy vyšší hodnotu průměrné délky cesty. Platí to pro všechny analyzované funkce ve všech modelech. Toto je patrné jak z aritmetických průměrů uvedených v tabulce 7, tak z většiny histogramů. Ukázka histogramu je na obrázku 13.

Nejnižší aritmetický průměr ve všech modelech měla vždy multimodální funkce a nejvyšší unimodální funkce.



Obrázek 13: Composite function 2 - IRM - Průměrná délka cesty

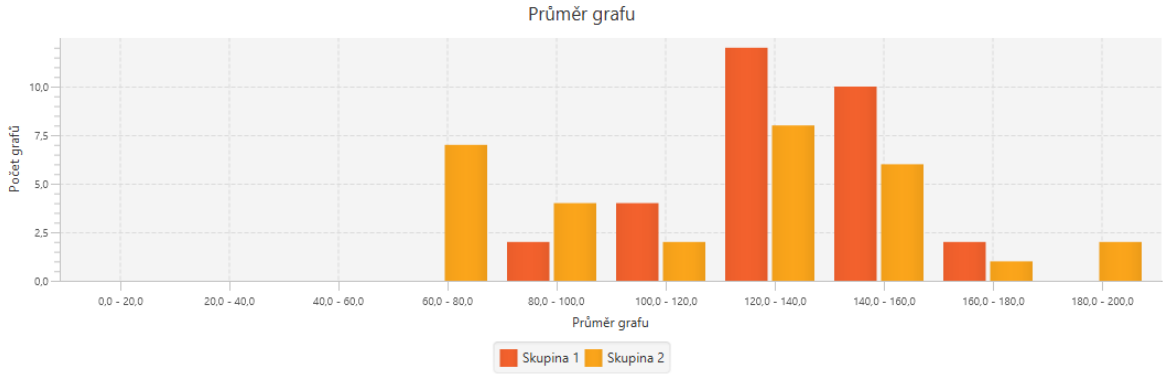
Model	Funkce	Průměr - lepší skupina	Průměr - horší skupina
INM	F01	1,0174	1,0015
	F10	0,4699	0,395
	F20	0,8215	0,7308
	F24	0,8646	0,6003
IVM	F01	1,4479	1,4104
	F10	0,5885	0,4856
	F20	1,092	0,9397
	F24	1,1677	0,7798
MM	F01	1,6237	1,5961
	F10	0,6097	0,4970
	F20	1,1677	1,009
	F24	1,1866	0,7915
IRM	F01	11,7168	11,0884
	F10	3,8312	1,1464
	F20	9,0134	7,1626
	F24	7,4883	5,3081

Tabulka 7: Průměrná délka cesty - Aritmetický průměr

11.3 Průměr grafu

Aritmetický průměr průměru grafu u modelu INM je ve všech analyzovaných funkcí roven 2. Modely IVM a MM nevykazují v aritmetickém průměru průměru grafu velký rozdíl. V modelu IVM měly funkce Hybrid function 4 a Composite function 2 aritmetický průměr v případě úspěšnějších běhů vyšší a funkce Rotated High Conditioned Elliptic Function a Shifted Schwefel's Function naopak nižší. V modelu MM pak měla funkce Rotated High Conditioned Elliptic Function aritmetický průměr v případě úspěšnějších běhů vyšší a ostatní funkce naopak

nižší. Modelem s výrazným rozdílem v této metrice je IRM. V tomto modelu měly všechny funkce aritmetický průměr v případě úspěšnějších běhů vyšší, než u běhů méně úspěšných. Z histogramů to na první pohled viditelné moc není, ale z aritmetických průměrů uvedených v tabulce 8 ano. Z toho vyplývá, že pokud v algoritmu SOMA dochází k častějším výměnám vedoucího jedince, tak algoritmus dosáhne lepšího řešení. Na obrázku 14 je ukázka histogramu této metriky.



Obrázek 14: Hybrid function 4 - IRM - Průměr grafu

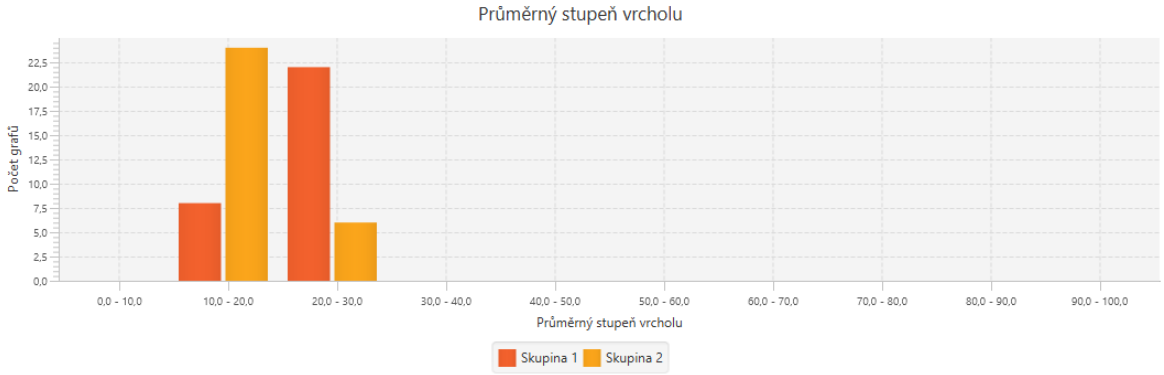
Funkce	**Průměr - lepší skupina**	**Průměr - horší skupina**
F01	115,2000	111,4667
F10	137,0667	46,0000
F20	134,1000	118,0667
F24	96,7000	84,8667

Tabulka 8: IRM - Průměr grafu - Aritmetický průměr

11.4 Průměrný stupeň vrcholu

Průměrný stupeň vrcholu opět ukazuje významný rozdíl mezi úspěšnými a méně úspěšnými běhy ve všech analyzovaných funkcích v modelu INM. Úspěšnější běhy mají vyšší průměrný stupeň vrcholu než běhy méně úspěšné. Průměrný stupeň vrcholu je v modelech IVM a MM stejný, jako v modelu INM. V modelu IRM je rozdíl velmi malý. Rozdíl je patrný z aritmetických průměrů uvedených v tabulce 9 a v případě modelu INM také z histogramu. Na obrázku 15 je pro ukázku zobrazen histogram této metriky. Z toho vyplývá, že během běhu algoritmu SOMA je lepší jedince vylepšit pomocí více různých jedinců. To je, jinými slovy řečeno, mít během běhu více různých vedoucích jedinců, což nám ukázala také analýza průměru grafu.

V modelu INM měla nejnižší aritmetický průměr hodnot multimodální funkce a nejvyšší unimodální funkce.



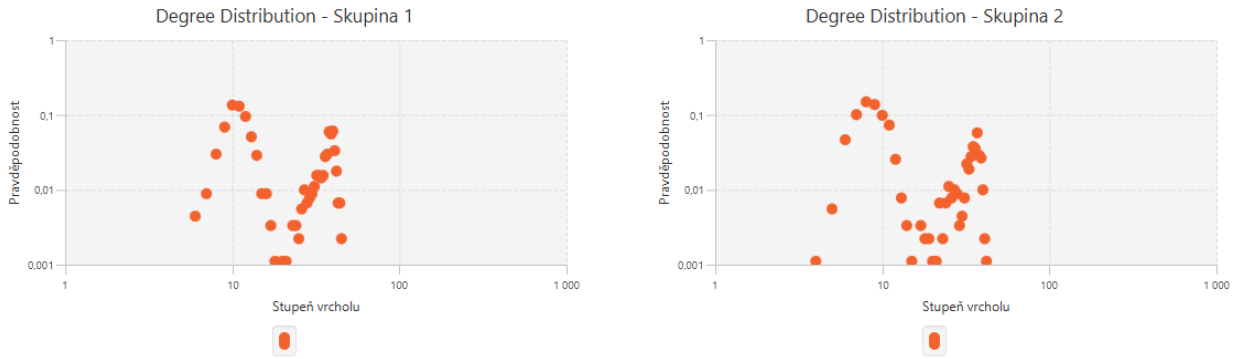
Obrázek 15: Shifted Schwefel's Function - INM - Průměrný stupeň vrcholu

Model	Funkce	Průměr - lepší skupina	Průměr - horší skupina
INM	F01	47,7089	45,4111
	F10	21,8511	17,4333
	F20	39,3644	35,7222
	F24	29,6133	22,2133
IRM	F01	3,9627	3,9608
	F10	3,9558	3,8717
	F20	3,9565	3,9485
	F24	3,9318	3,9169

Tabulka 9: Průměrný stupeň vrcholu - Aritmetický průměr

11.5 Degree distribution

Degree distribution je napříč funkcemi i kvalitami běhů velmi podobné. Zajímavé je, že v případě INM (stejně jako u IVM a MM) se mají vrcholy ve skupině tendenci shlukovat kolem dvou hodnot stupňů vrcholů. Tento jev, zobrazený na obrázku 16, se více či méně výrazněji projevuje u všech analyzovaných funkcí.

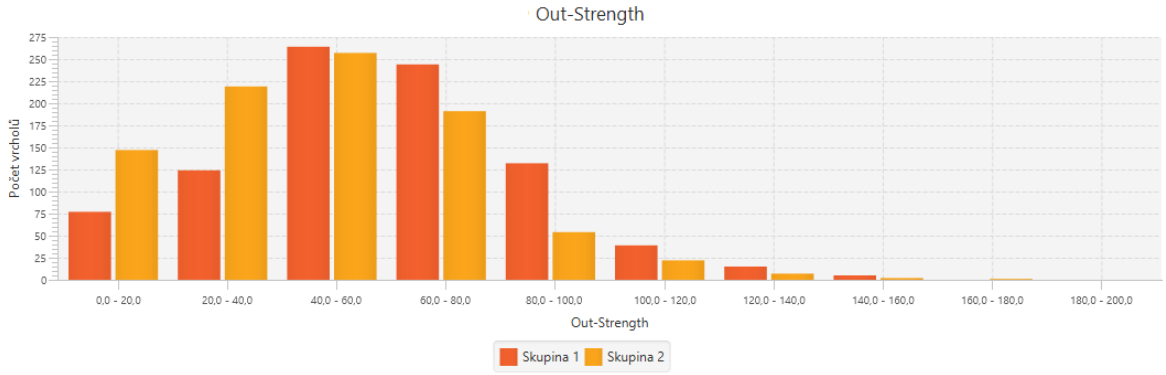


Obrázek 16: Shifted Schwefel's Function - INM - Degree distribution

11.6 Strength

Úspěšnější běhy jak v případě In-Strength, tak v případě Out-Strength dosahují vyšších hodnot, než v méně úspěšných bězích a to ve všech modelech i analyzovaných funkcích. Dobře patrné je to z aritmetických průměrů uvedených v tabulce 10 (In-Strength má stejný průměr jako Out-Strength). Z histogramů to vždy tak dobře viditelné není. Na obrázku 17 je ukázka histogramu Out-Strength. Z toho vyplývá, že čím více interakcí mezi jedinci v algoritmu SOMA, při kterých dojde k vylepšení jedince, tím lépe.

Nejvyšší hodnotu aritmetického průměru ve všech modelech měla vždy unimodální funkce. Nejnižší hodnotu aritmetického průměru v modelu INM měla multimodální funkce. V ostatních modelech měla nejnižší hodnotu v případě lepší skupiny kompozitní funkce a v případě horší skupiny multimodální funkce.



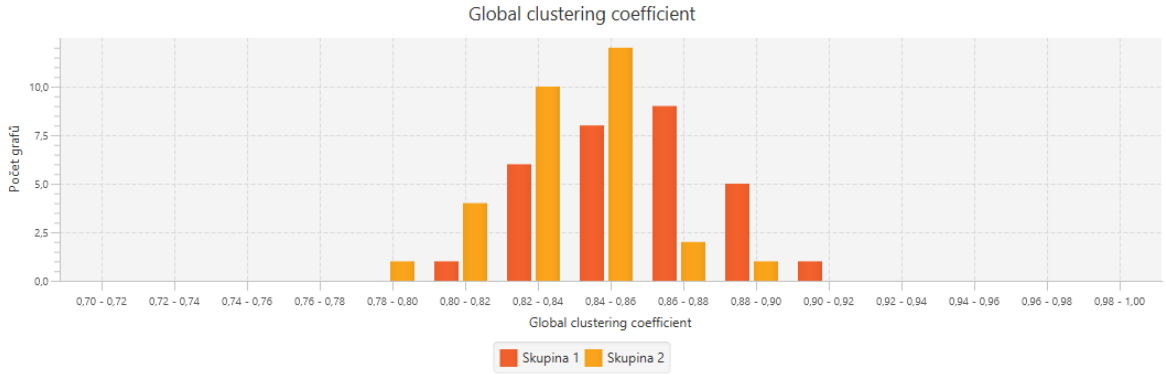
Obrázek 17: Composite function 2 - IVM - Out-Strength

Model	Funkce	Průměr - lepší skupina	Průměr - horší skupina
INM	F01	23,8544	22,7056
	F10	10,9256	8,7167
	F20	19,6822	17,8611
	F24	14,8067	11,1067
IVM	F01	107,5789	102,4322
	F10	96,3067	31,8544
	F20	95,3689	84,6611
	F24	59,18	47,8078
MM	F01	28,6929	25,8427
	F10	17,9313	2,6704
	F20	22,0981	17,5028
	F24	9,4575	6,5632
IRM	F01	1,9816	1,9807
	F10	1,9794	1,9391
	F20	1,9792	1,9767
	F24	1,9668	1,959

Tabulka 10: Strength - Aritmetický průměr

11.7 Clustering coefficient

Lokální i globální clustering coefficient je u modelu INM (stejně jako u IVM a MM), v případě lepších běhů, vždy mírně vyšší. Toto platí u všech analyzovaných funkcí. Je to dobře patrné u aritmetických průměrů, které jsou uvedeny v tabulce 11. Aritmetický průměr vychází stejně jak pro globální, tak pro lokální clustering coefficient. Tento malý rozdíl lze v histogramu špatně vidět. U globálního clustering coefficientu to lze v histogramu vidět lépe, než v případě lokálního. Ukázka histogramu je na obrázku 18. U všech analyzovaných funkcí je globální clustering coefficient vždy nad 0,7. Nejvyšších průměrných hodnot dosahuje unimodální a hybridní funkce. Z toho vyplývá, že se v grafu často objevují trojúhelníky a v algoritmu často dochází k situaci, kdy se jedinec x_i vylepšil o jedince x_j a x_k a zároveň se jedinec x_j také vylepšil o jedince x_k. Také z toho vyplývá, že čím častěji k této situaci dochází, tím je to pro výsledek algoritmu lepší.



Obrázek 18: Composite function 2 - INM - Globální clustering coefficient

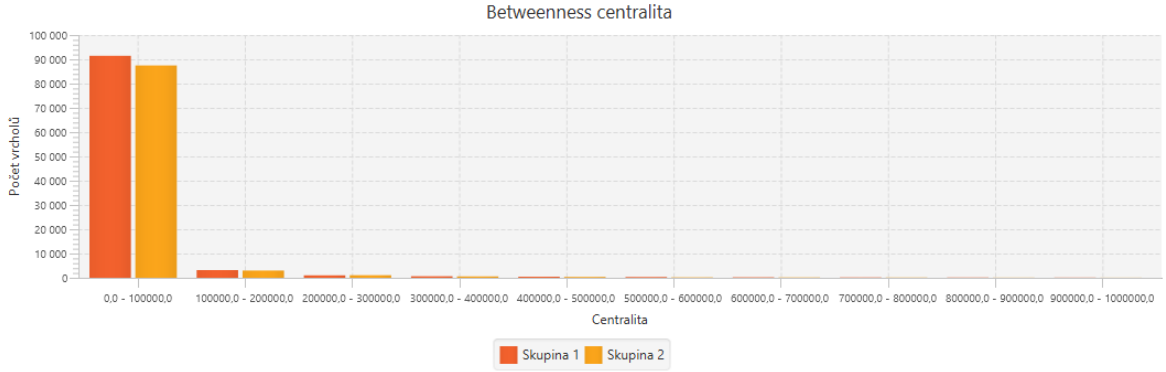
Funkce	Průměr - lepší skupina	Průměr - horší skupina
F01	0,9683	0,9552
F10	0,8429	0,8355
F20	0,9275	0,9063
F24	0,8597	0,8383

Tabulka 11: INM - Clustering coefficient - Aritmetický průměr

11.8 Betweenness centralita

Ukázku histogramu betweenness centrality můžeme vidět na obrázku 19. Významný rozdíl mezi lepšími a horšími běhy napříč funkcemi je v případě této metriky u IRM, pokud se podíváme na aritmetický průměr hodnot. V případě IRM mají běhy s lépe nalezeným řešením vždy vyšší průměrnou betweenness než běhy s hůře nalezeným řešením a to u všech analyzovaných funkcí. Aritmetické průměry u IRM jsou uvedeny v tabulce 12.

Nejvyšších průměrných hodnot dosahuje v modelu IRM unimodální funkce a nejnižších multimodální funkce.



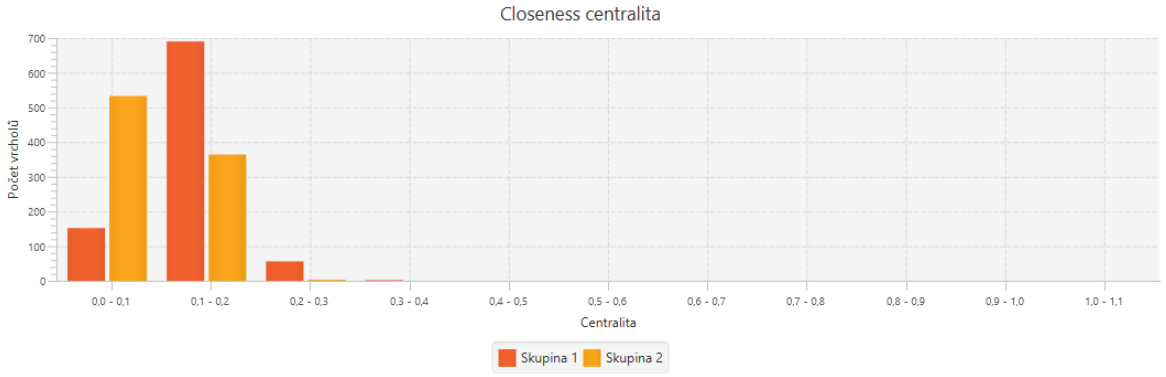
Obrázek 19: Rotated High Conditioned Elliptic Function - IRM - Betweenness centralita

Funkce	Průměr - lepší skupina	Průměr - horší skupina
F01	39010,4125	34992,7869
F10	12239,0256	1688,2524
F20	28285,5466	23521,708
F24	14594,6212	7767,5654

Tabulka 12: IRM - Betweenness centralita - Aritmetický průměr

11.9 Closeness centralita

Closeness centralita je v případě modelu MM u lepších běhů nepatrně vyšší než u běhů horších. Je to dobře viditelné jak z histogramů, tak z aritmetických průměrů uvedených v tabulce 13. Výjimkou je unimodální funkce, kde toto neplatí. Tato funkce má také nejvyšší průměrné hodnoty. Ukázka histogramu je na obrázku 20.



Obrázek 20: Shifted Schwefel's Function - MM - Closeness centralita

Funkce	Průměr - lepší skupina	Průměr - horší skupina
F01	0,4884	0,5678
F10	0,1242	0,0978
F20	0,1814	0,1546
F24	0,1832	0,1496

Tabulka 13: MM - Closeness centralita - Aritmetický průměr

11.10 Eigenvector centralita

Na základě analyzovaných funkcí není mezi touto centralitou a kvalitou nalezeného řešení žádný vztah v žádném modelu. Hodnoty byly ve všech funkcích i běžích obvykle do 0,1 výjimečně pak do 0,2. Aritmetický průměr hodnot byl vždy do 0,04.

12 Závěr

Pomocí grafů můžeme znázornit a analyzovat procesy z různých oblastí. V této práci jsme graf použili pro analýzu algoritmu SOMA. Tento algoritmus, který se řadí do skupiny biologicky inspirovaných výpočtů, jsem naimplementoval a spouštěl ho na testovacích funkcích CEC 2014. Během hledání řešení nad funkcí byl vždy vytvořen log soubor, který popisoval konkrétní běh algoritmu. Log soubory jsem následně rozdělil na dvě skupiny - úspěšnější běhy a méně úspěšné běhy. Dále jsem naimplementoval program pro analýzu grafů. Navrhl jsem do něj několik různých převodů běhu algoritmu SOMA na graf a s využitím knihovny Jung jsem do tohoto programu naimplementoval několik různých metrik. Grafy vytvořené z log souborů jsem následně vzájemně porovnával napříč různými funkcemi a různě kvalitním řešením pomocí tohoto programu. Cílem bylo zjistit, zda se metriky z různě dobrých běhů v něčem liší a zda by se to dalo využít pro vylepšení algoritmu, případně pro detekci dobrých a špatných řešení.

Z výsledků, nad námi vybranými daty, se ukázalo, že některé hodnoty metrik napříč různými funkcemi jsou stejné a jiné zase rozdílné. Analýza byla z důvodu velké časové náročnosti prováděna pouze na vzorku 4 funkcí. V budoucnu by bylo vhodné provést analýzu také na ostatních funkcích, aby se dali poznatky generalizovat. Nicméně ve všech čtyřech funkcích jsme až na pár výjimek popsaných v práci, co se týká lepších a horších řešení, dospěli ke stejným výsledkům. Z vybraných 4 funkcí jsme dospěli k následujícím zjištěním.

Assortative Mixing, Eigenvector centralita a Degree distribution byly napříč různými funkcemi a modely vždy prakticky totožné. Vypadá to, že tyto metriky nemají na výsledek algoritmu žádný vliv. O algoritmu SOMA z nich vyplývá pouze to, že jedinci mají tendenci úspěšně interagovat s mírně rozdílnými jedinci. Dále jsme z Degree distribution vypozorovali, že vrcholy ve skupině (lepší řešení a horší řešení) se mají tendenci shlukovat kolem dvou hodnot stupňů vrcholů.

V ostatních metrikách, zmíněných v této práci, již rozdíly mezi lepším a horším řešením byly. Z těchto metrik jsme vypozorovali, že čím více je úspěšných interakcí mezi jedinci a čím více je interakcí jedince s různými jedinci, tím lepších výsledků algoritmus SOMA dosahuje. Dalším zajímavým poznatkem je, že čím častěji v algoritmu dochází ke změně vedoucího jedince, tím lepších výsledků algoritmus dosahuje. Posledním poznatkem je, že čím častěji v algoritmu dochází k situaci, kdy se jedinec x_i vylepšil o jedince x_j a x_k a zároveň se jedinec x_j také vylepšil o jedince x_k, tak opět tím lepších výsledků algoritmus dosahuje. Na základě těchto metrik lze mezi dvěma skupinami běhů identifikovat skupinu, kdy algoritmus dosáhl lepších výsledků.

V budoucnu by stálo za pokus na základě těchto informací modifikovat algoritmus SOMA tak, aby k těmto výše zmíněným situacím docházelo co nejčastěji, což by mohlo vést ke zkvalitnění řešení nalezených algoritmem SOMA.

Literatura

[1] Cec 2014. `www.ntu.edu.sg/home/EPNSugan/index_files/CEC2014/CEC2014.htm`. Dostupné: 23.3.2018.

[2] Java. `www.java.com`. Dostupné: 23.3.2018.

[3] Jung - java universal network/graph framework. `www.jung.sourceforge.net`. Dostupné: 23.3.2018.

[4] Stefano Boccaletti, Vito Latora, Yamir Moreno, Martin Chavez, and D-U Hwang. Complex networks: Structure and dynamics. *Physics reports*, 424(4):175–308, 2006.

[5] Petr Burian. Využití evolučních výpočetních technik v elektronickỳch systémech–přehled zajímavỳch aplikací. *Elektrotechnika a informatika 2011. Část 2., Elektronika*, pages 13–16.

[6] Emre Cakir, Toni Heittola, Heikki Huttunen, and Tuomas Virtanen. Polyphonic sound event detection using multi label deep neural networks. In *Neural Networks (IJCNN), 2015 International Joint Conference on*, pages 1–7. IEEE, 2015.

[7] Paolo Crucitti, Vito Latora, and Sergio Porta. Centrality measures in spatial networks of urban streets. *Physical Review E*, 73(3):036125, 2006.

[8] Charles Darwin and Sir Gavin De Beer. The origin of species by means of natural selection: or, the preservation of favoured races in the struggle for life. Technical report, Oxford University Press, 1956.

[9] Dipankar Dasgupta. Advances in artificial immune systems. *IEEE computational intelligence magazine*, 1(4):40–49, 2006.

[10] Daniel Delling, Peter Sanders, Dominik Schultes, and Dorothea Wagner. Engineering route planning algorithms. In *Algorithmics of large and complex networks*, pages 117–139. Springer, 2009.

[11] Robert D Dony and Simon Haykin. Neural network approaches to image compression. *Proceedings of the IEEE*, 83(2):288–303, 1995.

[12] Paul K Harmer, Paul D Williams, Gregg H Gunsch, and Gary B Lamont. An artificial immune system architecture for computer security applications. *IEEE transactions on evolutionary computation*, 6(3):252–280, 2002.

[13] Petr Kovář. Úvod do teorie grafů. *Vysoká škola báňská. Technická univerzita. Ostrava*, 2012.

[14] JJ Liang, BY Qu, and PN Suganthan. Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization. *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore*, 2013.

[15] Norman Margolus, Tommaso Toffoli, and Gerard Vichniac. Cellular-automata supercomputers for fluid-dynamics modeling. *Physical Review Letters*, 56(16):1694, 1986.

[16] Radomír Měch and Przemyslaw Prusinkiewicz. Visual models of plants interacting with their environment. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 397–410. ACM, 1996.

[17] Mark Newman. *Networks: an introduction*. Oxford university press, 2010.

[18] Mark EJ Newman. Scientific collaboration networks. ii. shortest paths, weighted networks, and centrality. *Physical review E*, 64(1):016132, 2001.

[19] Mark EJ Newman. Mixing patterns in networks. *Physical Review E*, 67(2):026126, 2003.

[20] Mark EJ Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003.

[21] Gabriela Ochoa. On genetic algorithms and lindenmayer systems. In *International Conference on Parallel Problem Solving from Nature*, pages 335–344. Springer, 1998.

[22] Giuliano Andrea Pagani and Marco Aiello. The power grid as a complex network: a survey. *Physica A: Statistical Mechanics and its Applications*, 392(11):2688–2700, 2013.

[23] Camelia-Mihaela Pintea. Bio-inspired computing. In *Advances in Bio-inspired Computing for Combinatorial Optimization Problems*, pages 3–19. Springer, 2014.

[24] Sergio Porta, Paolo Crucitti, and Vito Latora. The network analysis of urban streets: a dual approach. *Physica A: Statistical Mechanics and its Applications*, 369(2):853–866, 2006.

[25] Lukas Tomaszek and Ivan Zelinka. On performance improvement of the soma swarm based algorithm and its complex network duality. In *Evolutionary Computation (CEC), 2016 IEEE Congress on*, pages 4494–4500. IEEE, 2016.

[26] Lukáš Tomaszek and Ivan Zelinka. Conversion of soma algorithm into complex networks. In *Evolutionary Algorithms, Swarm Dynamics and Complex Networks*, pages 101–114. Springer, 2018.

[27] Johan Ugander, Brian Karrer, Lars Backstrom, and Cameron Marlow. The anatomy of the facebook social graph. *arXiv preprint arXiv:1111.4503*, 2011.

[28] Xiao Fan Wang and Guanrong Chen. Complex networks: small-world, scale-free and beyond. *IEEE circuits and systems magazine*, 3(1):6–20, 2003.

[29] Daan Wierstra, Tom Schaul, Jan Peters, and Juergen Schmidhuber. Natural evolution strategies. In *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 3381–3387. IEEE, 2008.

[30] Stephen Wolfram. Cellular automata as models of complexity. *Nature*, 311(5985):419–424, 1984.

[31] Xin-She Yang. *Nature-inspired optimization algorithms*. Elsevier, 2014.

[32] Ivan Zelinka and Guanrong Chen. *Evolutionary Algorithms, Swarm Dynamics and Complex Networks: Methodology, Perspectives and Implementation*, volume 26. Springer, 2017.

[33] Ivan Zelinka, Zuzana Oplatková, Pavel Ošmera, Miloš Šeda, and František Včelař. *Evoluční výpočetní techniky*. BEN, 2008.

[34] Ivan Zelinka, Lukas Tomaszek, and Lumir Kojecky. On evolutionary dynamics modeled by ant algorithm. In *Intelligent Networking and Collaborative Systems (INCoS), 2016 International Conference on*, pages 193–198. IEEE, 2016.

[35] Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling neural networks: many could be better than all. *Artificial intelligence*, 137(1):239–263, 2002.